CHAPTER 8

COMPUTER INSTRUCTIONS AND MAN/MACHINE INTERFACES

INTRODUCTION

You have probably heard people talk about the various computer programming languages used to write computer programs. Maybe you have even written some. Programming languages include procedural-type languages. Examples are COBOL (COmmon Business Oriented Language) to solve business-type problems, and FORTRAN (FORmula Translation) to solve mathematical-type problems. Other languages are interactive languages that enable a person to communicate with a computer in a conversational mode to develop programs. BASIC (Beginner's All-Purpose Symbolic Instruction Code) is an example of an interactive language. Another language called Ada is the language developed for the Department of Defense for use in embedded applications; for example, where a computer serves as a control system. (Ada is named for Ada Augusta Byron, Countess of Lovelace, for her achievements relating to computers. She was a full collaborator and suggested the use of the binary system rather than the decimal system to Charles Babbage, who is recognized as the father of computers.) These are all considered high-level programming languages in that their instructions are in human readable form, such as ADD A to B; LET X = Y, IF A > Y, THEN PRINT Y, and so on. These types of instructions must be translated into machine code for execution by a computer. This is accomplished through special language translation programs. For high-level languages, a compiler program maybe used.

There are two other levels of computer languages: assembly language and machine language. Assembly languages use mnemonics, symbols, to represent operations. For example, "A" might mean add and "STR" might mean store. Like high-level languages, these must be translated before a computer can execute the instructions. To translate assembly language programs, an assembler program is used.

By now, you have probably noticed that for an instruction to be executed, it must be in machine code that consists of a series of 0's and 1's—the only things a computer can understand. You have probably also realized that to write instructions in 0's and 1's would be tedious, difficult, and time consuming. Therefore, the assembly languages and the high-level languages provide easier means for people to use to interface with computers to specify the steps a computer is to perform. As a technician who is looking primarily at the internal functions of a computer, you need to understand machine code and how it works. Some of the operator/maintenance panels display information in binary, as you have already learned. It will be up to you to interpret codes as meaningful information. Other displays present information in commonly used words, terms, and numbers. In these cases a computer, through program instructions, translates/interprets the binary codes into meaningful information. This information is then presented to you.

As we have just said, the machine instructions (code) provide the computer with the means to carry out various operations; both internal and external. Internal and external operations include processing the data and interfacing with other computers, peripherals, and display and communications systems as part of a computer system and performing maintenance.

The man/machine interfaces enable you to communicate with the computer's hardware and software through controlling devices and software/programs.

After completing this chapter, you should be able to:

- **Describe and recognize instruction types and their uses**
- **Describe the types of instructions, their designators and classes, used by computers**
- **Describe how to interface with a computer's hardware and software**

We begin by discussing computer instructions; program types; and instruction levels, types, interpretation, formats, sizes, and operand addressing.

TOPIC 1—COMPUTER INSTRUCTIONS

Computer instructions tell the equipment to perform a designated operation. These machine instructions are contained in an instruction set (the **computer's repertoire of instructions).** They will be processed by the CPU. Some computers have an I/O controller (IOC) unit with its own set of instructions. Each instruction in the set/repertoire contains at least an operation (op) code to tell the CPU what operation to peform. It may also contain an operand to identify the address part of the instruction and/or other information (designators) needed by the CPU to perform the operation.

Before we discuss individual instruction types and formats, let's look at some of the types of computer programs/software commonly used.

TYPES OF COMPUTER PROGRAMS/SOFTWARE

A computer program is a sequence of instructions, written in a specified way to perform a **plan** (an algorithm) and/or **routine.** Programs are written to manage a computer and its resources, solve a problem or type of problem, and/or diagnose malfunctions in a computer. Programs include **hardwired** (read-only) programs stored in a read-only memory (ROM) or programmable ROM (PROM). They also include programs that were written by **programmers** and can be altered (authorized software changes) as required. Hardwired programs are installed at the factory and cannot be altered except by replacing the ROM or reprogramming the PROM. Other programs are generally stored on magnetic media (tape or disk) or on optical compact disk (CD) ROM. These programs are loaded into computer memory when needed.

You will encounter several general types of programs. These include operating systems, application/operational programs, and utility programs (utilities). Programs may be written to run on a stand-alone computer or interactively on two or more computers connected together.

Operating Systems

An operating system is a collection of many programs used by a computer to manage its own resources and operations. The types of operating systems include the following:

- Single tasking
- Multitasking
- Real-time
- Local-area network
- Wide-area network

- Virtual (VOS)
- Disk (DOS)

Operating systems provide the link between the hardware and its user as well as enabling the execution of operational and/or application programs designed for specific use.

Application/Operational Programs

Programs for the computers you maintain will be used in tactical, tactical support, and nontactical platforms. These programs are designed to solve specific types of problems. They are commonly called application programs, operational programs, or processing programs. The programs used in tactical or tactical support platforms, such as CDS/NTDS or ASWOCs, are generally called operational programs. The programs used with the SNAP systems (I and II) are, as a rule, called application programs. Programs available commercially that are designed to solve specific classes of problems are often called packaged software or off-the-shelf software. These include word processing, database management, graphics, spreadsheet, and desktop publishing programs to name a few.

Utility Programs

Utility programs include general routines or diagnostics run by the computer to test other equipments or itself. A programmed operational and functional appraisal (POFA) to test magnetic tape units and a diagnostic test for a computer are examples. Utilities can be run as stand-alone programs, such as microcomputer diagnostics, a maintenance test program (MTP), a POFA, and a standard test program (STP) using a standard test driver (STD). They can also be run as part of an operating system (if memory permits) or as online diagnostic tests such as on a SNAP system or NTDS. Utility programs also include programs and routines to perform general routine tasks, such as disk/tape copy and print. These, too, can be stand-alone programs or they maybe included with the operating system or other programs.

LEVELS OF INSTRUCTIONS

The CPU executes machine instructions, which manipulate the data within the functional units of the computer. In early computers, only one level of machine instructions was used. In modern computers, this only remains true in microprocessors and most microcomputers. For most computers, there are now two levels of machine instructions: **microinstruction** and **macroinstructions.** In larger microprocessor-based devices (minicomputers and mainframes), each microinstruction is in effect a predetermined and installed set of microinstruction.

The particular device's **instruction set** is made up of the highest levels (micro or macro) of machine instructions. The instruction set is the complete set of individual operations that can be executed or performed by the particular microprocessor or computer. In microprocessors, microcomputers, and microprocessor controlled peripherals, the machine instructions are referred to as microinstruction, and the microprocessor executes them to perform the desired operations.

In mini and mainframe computers, the machine instructions are actually macroinstructions. Once again, a microinstruction is a predetermined or preset sequence of microinstruction. Since most of the larger devices are microprocessor driven, it is necessary to break down the larger microinstruction into a series of smaller events that a microprocessor can handle. The microinstruction that make up the macroinstructions do not normally concern the computer programmer who uses only the microinstruction set. The microinstruction are usually stored in some form of local memory, accessible only to the microprocessor translating and executing the macroinstructions.

Instruction sets differ to some degree between computers, particularly between those of different manufacturers, types, and generations of computers. The actual number of instructions in an instruction set has a direct affect on the overall operation of the device. Computers with small instruction sets are easier to understand, and this simplifies both programming and maintenance. A large instruction set tends to support more specialized activities or functions that make the overall operation of the device more efficient or more tailored to the user's requirements. An example of a large instruction set is one used on large mainframes aboard a ship.

TYPES OF INSTRUCTIONS

The flow of data in a computer is the result of instruction execution. Data can be exchanged between registers. It can be moved from one register to another. It can be moved from a register to a memory location or vice versa. Arithmetic instructions can be performed using the contents of registers and memory locations. Logical instructions can be used to isolate bits in registers and memory locations. How machine instructions within an instruction set are classified differ by computer type and manufacturer.

Instructions Classified by Function

Instructions can be classified in general terms by the type of operation they peform—data movement, transfer of control or program sequencing, arithmetic, and logical.

MOVEMENT INSTRUCTIONS.— Movement instructions literally move the data in some way. They include internal, external, and data assignment instructions.

Internal Instructions.—Internal instructions move the data within the confines of the computer. They include the following examples:

- Load —Load the A register with the contents specified by the operand
- Move (transfer) —Move the contents of one register to another register
- Store —Store the contents of a register into a specified memory address

External Instructions.—Instructions dedicated to I/O are external instructions. They include **inputting** data from a peripheral device, such as a magnetic tape unit or **outputting** data to a peripheral, such as a printer.

Data Assignment (Special-Purpose) Instructions.— Data assignment instructions include those that set or clear status indicating bits that are normally held in an active status or flag register. Some examples of active status registers include state indicators, upper/lower control indicator of half-word instructions, interrupt lockouts, memory lockout inhibit, bootstrap mode, fixed point overflow, and compare designators. Some examples of flag bits are equal to zero, sign (+ or –), carry, and parity (odd or even).

TRANSFER OF CONTROL OR PROGRAM SEQUENCING CONTROL INSTRUCTIONS.— Transfer of control or program sequencing control instructions enable the programmer to change the sequence in which instructions are executed by branching to another area of a program. They also include instructions for a **subroutine** to perform a function.

Branching Instructions.— Branching instructions make it possible to change the sequence in which the computer peforms instructions. An **unconditional** branching instruction always causes a jump to a new area of memory. An example is a jump (JMP) instruction. Branching instructions often use a modifier in the instruction to establish a condition to be met. These types of branching instructions are called conditional branching instructions. A **conditional** branching instruction causes a jump to a new area of memory only when a specific condition is met, such as IF A = 0 JUMP TO.... A conditional branching instruction may also rely on the setting of a switch on the computer's controlling device to be included with the instruction, such as IF JUMP 1 SWITCH IS SET JMP TO . . . .

Subroutine.— A subroutine may include a function that is routinely repeated, such as incrementing or decrementing an index register or a short multiply routine when no multiply instruction exists. Some functions performed in a subroutine may include **stack pointer** management and data buffering algorithms such as **last-in, first-out (LIFO)** and **first-in, first-out (FIFO)** methods.

ARITHMETIC INSTRUCTIONS.— Arithmetic instructions include add, subtract, multiply, divide, shift, increment, decrement, clear, and negation instructions. Depending on the design of the computer, absolute numbers are involved in arithmetic calculations. Also depending on the design, **math pac** and **numeric data coprocessor** are used in some computers in addition to the normal arithmetic instructions available. They execute the arithmetic instructions the CPU's ALU cannot and are still controlled by the CPU's program control.

LOGICAL INSTRUCTIONS.— Logical instructions include and, or, not, exclusive or/nor, compares, and shift instructions. They are often used in computers with multiply or divide instructions, in calculations to isolate bits. They also include compare type instructions. Compare type instructions are greater than (>), less than (<), equal to (=), not equal to (<>), check for positive, and check for negative.

Instructions Classified by Their Action on Operands

Instructions may also be classified by their action on an operand. They may read, store, or replace an operand. For example, ADD LOGICAL PRODUCT is classified as a read instruction; STORE LOGICAL PRODUCT is classified as a store instruction; and REPLACE SELECTIVE CLEAR is classified as a replace instruction. Figure 8-1 illustrates instructions with their classification (read (R), store (S), or replace (RP)).

- Read —Read instructions acquire an operand from main memory.

- Store —Store instructions process an operand already acquired and store it in main memory.

- Replace —Replace instructions acquire and process an operand and then store it in memory.

INSTRUCTION LANGUAGE INTERPRETATION

The instruction format provides the means to customize each instruction. A list of instructions with their formats, symbols, and meanings provides you a means to interpret what an instruction will ultimately accomplish. This is a very useful troubleshooting tool to help isolate a specific malfunction. The instructions differ between types of computers. Take a

FUNCTION (OP) CODE	NAME	FORMAT	CLASS
0 0	Illegal	—	
01 0	Inclusive OR (Selective Set A)	II	R
01 1	Selective Clear A	II	R
01 2	Selective Substitute	II	R
01 3	Exclusive OR (Selective Complement A)	II	R
01 4	Add Logical Product	II	R
01 5	Load Logical Product	II	R
01 6	Subtract Logical Product	II	R
01 7	Load Logical Product Next	II	R
02 0	Count Ones	II	R
02 1	Illegal	—	—
02 2	Execute Remote	II	R
02 3	Execute Remote Lower	II	R
02 4	Store Logical Product	II	S
02 5	Store Sum	II	S
02 6	Store Difference	II	S
02 7	Double Store A	II	S
03 0	Replace Inclusive OR	II	RP
03 1	Replace Selective Clear	II	RP
03 2	Replace Selective Substitute	II	RP

Figure 8-1.—Examples of computer instructions.

few minutes to study figures 8-2, A and 8-2, B. Figure 8-2, A shows two examples of instructions used in tactical data systems. Figure 8-2, B shows examples of instructions used on a typical general-purpose microcomputer. By looking at the information provided about an instruction, you will be able to tell the **op (function)** in hex or octal, **operation or name, mnemonic,** and **description** or **Boolean/ arithmetic operation;** You will also notice the parts peculiar to a specific computer and its instructions. Among those are addressing modes, status indicating registers, coding format, and soon.

CPU

REPERTOIRE OF INSTRUCTIONS

CODE	Mnemonic	NAME	DESCRIPTION	F	CA	R	UF	TIME: #S
00	ILLEGAL							
01 0	OR	INCLUSIVE OR (SELECTIVE SET A)	$(Y) \oplus (A_a) \rightarrow A_a$	11	Y	Y	2	1.5
01 1	SC	SELECTIVE CLEAR A	$(Aa) \odot (Y)' \rightarrow Aa$	11	Y	Y	2	1.5
01 2	MS	SELECTIVE SUBSTITUTE	$(Y)_n \rightarrow (A_a+1)n$ FOR ALL $(A_a)_n = 1$; $(A_a)_i = (Aa)_f$	11	Y	Y	2	1.5
01 3	XOR	EXCLUSIVE OR (SEL. COMP. A)	$(Y) \bigoplus (A_a) \rightarrow A_a; (A_a)n' \rightarrow (A_a)n$ FOR $(Y)_n = 1$	11	Y	Y	2	1.5 1.5

CMR - CONTROL MEMORY REGISTER	UF - ULTRA FORMAT	$\underline{Y}$-OPERAND (Y) (WHOLE WORD OR PARTIAL WORD) OR Y, DEPENDING ON K
F - FORMAT	(A)n- CONTENTS OF A, BIT n	
CA - CHARACTER ADDRESSABLE	CD - COMPARE DESIGNATOR	$\odot$ - LOGICAL PRODUCT (AND)
R - REPEATABLE	Y - ADDRESS FORMED BY y + (Bb) + (Ss)	$\oplus$ - LOGICAL SUM (INCLUSIVE OR)
DSW - DESIGNATOR STORAGE WORD	ICW - INITIAL CONDITION WORD	$\bigoplus$ - LOGICAL DIFFERENCE (EXCLUSIVE OR)

CENTRAL PROCESSOR UNIT (CPU) INSTRUCTIONS

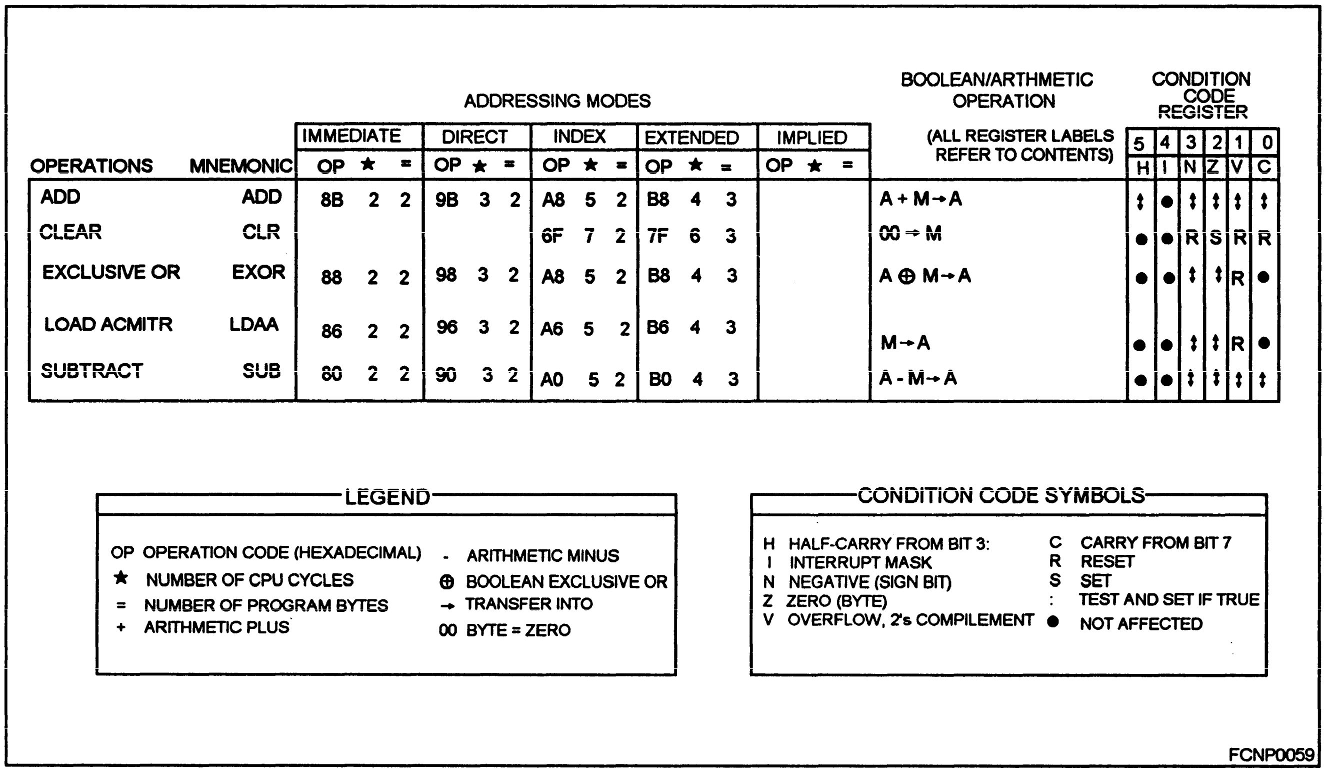
OCTAL FORMAT	CODING FORMAT		PRIV	7	NAME	DESCRIPTION	F	1A	CA	R	ISA REF
00	N/A				ILLEGAL	N/A					
01 0	OR	a,y,b,s	N	Y	INCLUSIVE OR (SELECTIVE SET A)	$(Y) \oplus (A_a) \rightarrow A_a$	11	Y	Y	Y	A.2
01 1	SC	a,y,b,s	N	Y	SELECTIVE CLEAR A	$(A_a) \odot (Y)' \rightarrow A_a$	11	Y	Y	Y	A.3
01 2	MS	a,y,b,s	N	Y	SELECTIVE SUBSTITUTE	$(Y)_n \rightarrow (A_a+1)_n$ FOR ALL $(A_a)_n = 1; (A_a)i = (A_a)_f$	11	Y	Y	Y	A.4
01 3	XOR	a,y,b,s	N	Y	EXCLUSIVE OR (SEL. COMP.A)	$(Y) \bigoplus (A_a) \rightarrow A_a; (A_a)_n' \rightarrow (A_a)_n$ FOR $(Y)_n = 1$	11	Y	Y	Y	A.5

SYMBOL	DEFINITION
a	INSTRUCTION FIELD DESIGNATING AN ACCUMULATOR REGISTER INDEX REGISTER. OR IOC NUMBER
b	INSTRUCTION FIELD DESIGNATING AN INDEX OR ACCUMULATOR REGISTER.
CA	CAPABLE OF BEING INDIRECT ADDRESSABLE: Y = CAPABLE N = NOT CAPABLE
F	INSTRUCTION FORMAT DESIGNATION
1A	CAPABLE OF BEING INDIRECT ADDRESSABLE: Y = CAPABLE OF INDIRECT ADDRESSING. N = NOT CAPABLE OF INDIRECT ADDRESSING. C = CAPABLE OF INDIRECT ADDRESSING UNDER CERTAIN CONDITIONS; REFER TO ISA REF COLUMN.
ISA REF	CONTAINS PARAGRAPH REFERENCES TO THE ISA SPECIFICATION TO FURTHER EXPLAIN INSTRUCTION OPERATION

SYMBOL	DEFINITION
PRIV	COLUMN DESIGNATES WHICH INSTRUCTIONS ARE PRIVILEGED: P = PRIVILEGED N = NOT PRIVILEGED C = PRIVILEGED UNDER SPECIAL CONDITIONS (REFER TO IS A REF COLUMN).
R	CAN BE REPEATED
s	INSTRUCTION FIELD DESIGNATING A BASE REGISTER
y	INSTRUCTION OPERAND FIELD DESIGNATING AN ADDRESS OR VALUE
Y	EFFECTIVE ADDRESS FORMED BY y + (Bb)+[(Ss) + (Qs)] (CMP MODE.) EFFECTIVE ADDRESS FORMED BY y + (Bb) + (Ss) (NTV MODE.)'
7	THIS COLUMN INDICATES WHICH INSTRUCTIONS ARE EXECUTABLE IN CMP MODE: Y = EXECUTABLE N = NOT EXECUTABLE D = THESE INSTRUCTIONS WHEN EXECUTED IN THE NTV MODE OPERATE DIFFERENTLY THAN IN THE CMP MODE. REFER TO THE PARAGRAPH NOTED IN THE ISA REF COLUMN FOR THE DIFFERENCES.

FCNP0058

Figure 8-2, A.—Examples of instruction interpretations for two mainframe computers.



		ADDRESSING MODES															BOOLEAN/ARTHMETIC OPERATION	CONDITION CODE REGISTER					
		IMMEDIATE			DIRECT			INDEX			EXTENDED			IMPLIED			(ALL REGISTER LABELS REFER TO CONTENTS)	5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	★	=	OP	★	=	OP	★	=	OP	★	=	OP	★	=		H	I	N	Z	V	C
ADD	ADD	8B	2	2	9B	3	2	A8	5	2	B8	4	3				A + M → A	↕	●	↕	↕	↕	↕
CLEAR	CLR							6F	7	2	7F	6	3				00 → M	●	●	R	S	R	R
EXCLUSIVE OR	EXOR	88	2	2	98	3	2	A8	5	2	B8	4	3				A ⊕ M → A	●	●	↕	↕	R	●
LOAD ACMITR	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	●	●	↕	↕	R	●
SUBTRACT	SUB	80	2	2	90	3	2	A0	5	2	B0	4	3				A - M → A	●	●	↕	↕	↕	↕

LEGEND

OP OPERATION CODE (HEXADECIMAL)	- ARITHMETIC MINUS
★ NUMBER OF CPU CYCLES	⊕ BOOLEAN EXCLUSIVE OR
= NUMBER OF PROGRAM BYTES	→ TRANSFER INTO
+ ARITHMETIC PLUS	00 BYTE = ZERO

CONDITION CODE SYMBOLS

H	HALF-CARRY FROM BIT 3:	C	CARRY FROM BIT 7
I	INTERRUPT MASK	R	RESET
N	NEGATIVE (SIGN BIT)	S	SET
Z	ZERO (BYTE)	:	TEST AND SET IF TRUE
V	OVERFLOW, 2's COMPILEMENT	●	NOT AFFECTED

FCNP0059

Figure 8-2, B.—An example of instructions for a typical microcomputer.

INSTRUCTION FORMATS

Instruction formats vary between microprocessors and minicomputers and mainframe computers. As the machine instructions are generally longer in larger computers with their larger memory words, the instruction format or how the instruction is translated differs. Each instruction is composed of fields. The lengths of instructions and the lengths and positions of the fields differ depending on the instruction and the computer. An operation (function) code is part of all instructions. How the remainder of the instruction is translated and the names assigned to the parts vary. Let's take a look at two examples of computer instruction formats, one for a microcomputer and one for a mainframe. We begin with the op (function) code, which is common to both; only the length differs.

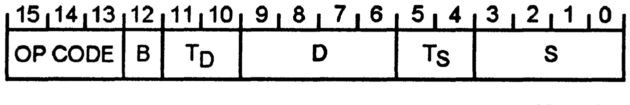
A typical machine instruction begins with the specification of an operation to be performed, the **operation (op) code.** Refer back to figure 8-1. The op code tells the computer/processor what basic operation to perform. The op code, a part of every instruction, is usually located at the beginning of each instruction format. Following the op code is information, if needed, to define the location of the data or the **operand** on which the operation is to be performed. This location in memory, called the **operand address,** normally defines the location that contains the operand at the start of the operation (the source), or that will contain the modified operand upon completion of the operation (the destination).

The remainder of the instruction and how it is structured differs from one computer or computer type to another. The **designators** in each field and the positions of the fields within the instruction determine how the instruction will affect the operand, registers, memory, and general flow of data in and out of the computer. We discuss the fields and the designators as we discuss the two instruction formats.

Microcomputer Instruction Formats

A basic 16-bit microinstruction is divided into a number of separate fields. Refer to figure 8-3 as a reference. You'll notice the lengths of the fields vary.

The op code is located in the most significant bits (2^{15} through 2^{13}). B (bit 2^{12}) tells the computer to use all 16 bits as a word or divide the 16 bits into 8-bit bytes.



35NVM059

Figure 8-3.—Example of microinstruction format.

D (bits 2^9 through 2^6) is a code identifying the destination portion of the instruction. S (bits 2^3 through 2^0) identifies the source portion. T_D (bits 2^{11} and 2^{10}) and T_S (bits 2^5 and 2^4) are bits in the instruction word that identify the type of addressing mode being used to locate the destination and source addresses.

As shown in figure 8-4, two or three memory words are required for some instructions depending on the addressing mode indicated by T_D and Ts. Addressing modes are discussed in the next section. Microcomputers may have more than one instruction format for the one word instructions. The format depends on the type of instruction being used.

Mainframe Computer Instruction Formats

The instruction formats for large mainframe computers vary greatly between types, generations, and manufacturers of computers. For our example, we selected the instruction format for CPU instructions of a mainframe computer with 32-bit computer instructions. These instructions can have up to seven basic formats designated I, II, III, IV-A, IV-B, IV-C and V. The majority of these instructions are full memory word (32-bit) instructions. Only formats IV-A, IV-B,

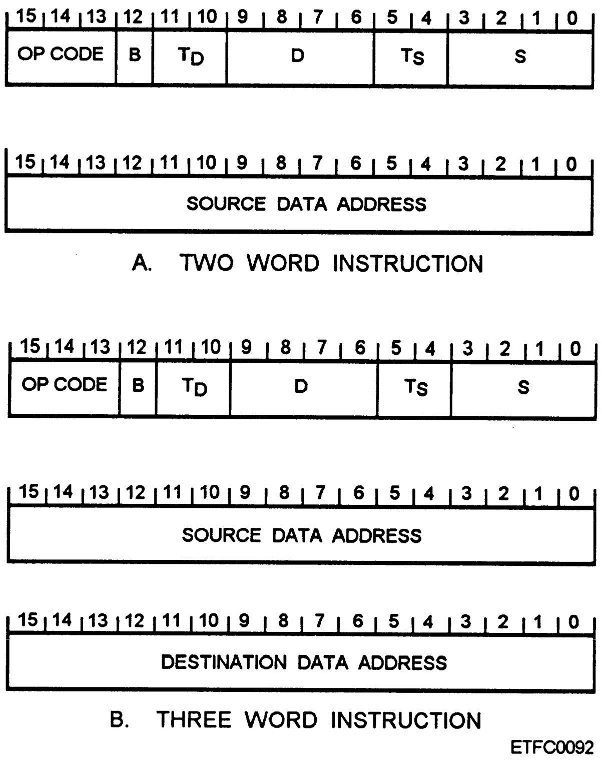


Figure 8-4.—Microcomputer instruction formats with two and three memory words.

and IV-C are upper or lower half-word (16-bit) instructions.

The instructions are divided into a number of single or multibit fields that each perform a specific function during instruction execution. Two fields called the function code **(f)** and the accumulator or index **(a)** designator fields are consistent throughout all the formats. The f field is the 6-bit function code (op code) and the a field is the 3-bit accumulator register designator field.

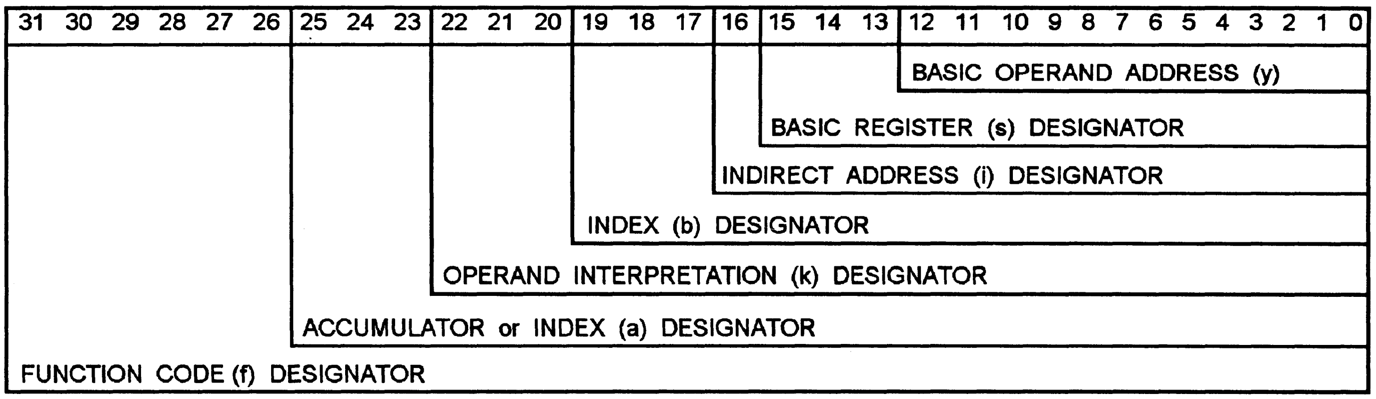
The function code **(f)** defines the complete operation to be performed or it may be used in conjunction with other fields called **subfunction designators** to define an operation. The accumulator register designator **(a)** field is used to identify the particular accumulator (0-7), index (0-7), or stack pointer register (0-7) needed for the operation. The formats and instruction fields are described in the following paragraphs.

● Formats I, II, and III —These three formats (fig. 8-5) make up the majority of instructions in the example computer's repertoire of instructions. Format I instructions perform the basic load, store, replace, and simple mathematical operations for the computer. Format II instructions are concerned with single precision mathematics, interrupt, and I/O commands. Format 111 instructions are used for program sequence control (jumps, return jumps, and switch controlled or manual jumps).

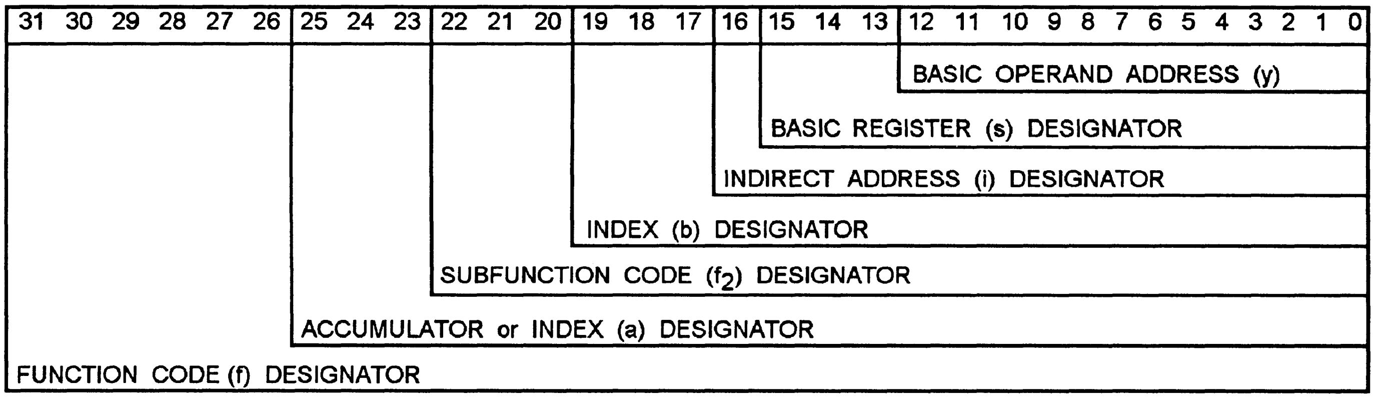
The three formats have many fields in common. The nine most significant bits (2^{31} through 2^{23}) are made up of the **f** and **a** fields. Only bits 2^{22} through 2^{20} differ between the three formats. In format I, the 3-bit field is called the **k** field or operand interpretation designator. This field is used primarily during mathematical operations. In format II instructions, the three bits become a subfunction code **(f_2).** And in format III instructions, the three bits become a two-bit subfunction code **(f_3)** and a single-bit **k** code that is always ZERO for format III instructions.

NOTE: Subfunction codes, f_2 through f_6, are used as part of the op code unless otherwise specified. A subfunction code of two bits has a maximum value of 3 (11_2). A subfunction code of three bits has a maximum value of 7 (111_2). For example, the format II op code 07 could have a subfunction 7 and format III op code 53 could have a subfunction code of 3.

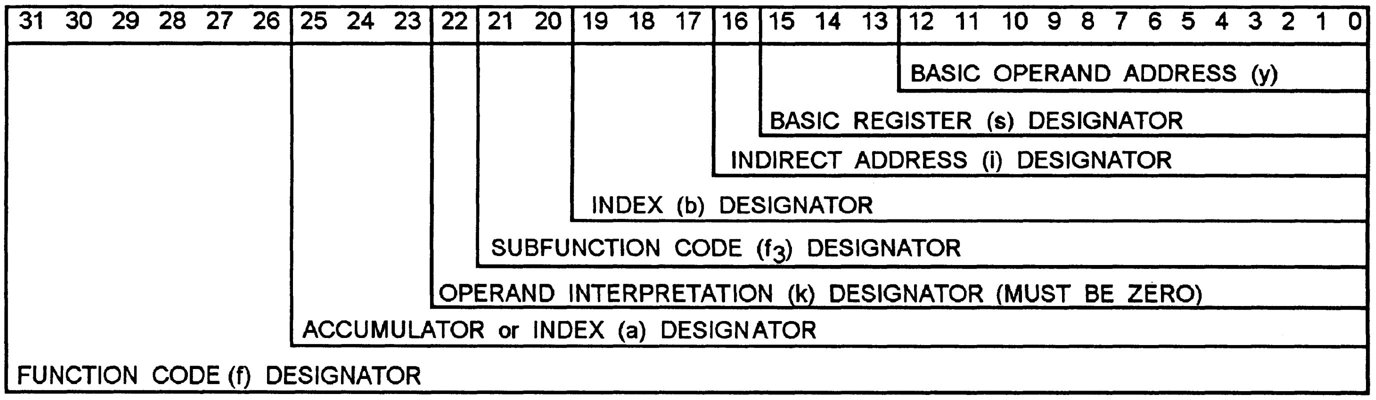
The remainder of the instruction, bits 2^{19} through 2^0, is the same for all three formats. There is a 3-bit index register designator code **(b),** a single-bit indirect


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
BASIC OPERAND ADDRESS (y)
BASIC REGISTER (s) DESIGNATOR
INDIRECT ADDRESS (i) DESIGNATOR
INDEX (b) DESIGNATOR
OPERAND INTERPRETATION (k) DESIGNATOR
ACCUMULATOR or INDEX (a) DESIGNATOR
FUNCTION CODE (f) DESIGNATOR

FORMAT I


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
BASIC OPERAND ADDRESS (y)
BASIC REGISTER (s) DESIGNATOR
INDIRECT ADDRESS (i) DESIGNATOR
INDEX (b) DESIGNATOR
SUBFUNCTION CODE (f_2) DESIGNATOR
ACCUMULATOR or INDEX (a) DESIGNATOR
FUNCTION CODE (f) DESIGNATOR

FORMAT II


31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
BASIC OPERAND ADDRESS (y)
BASIC REGISTER (s) DESIGNATOR
INDIRECT ADDRESS (i) DESIGNATOR
INDEX (b) DESIGNATOR
SUBFUNCTION CODE (f_3) DESIGNATOR
OPERAND INTERPRETATION (k) DESIGNATOR (MUST BE ZERO)
ACCUMULATOR or INDEX (a) DESIGNATOR
FUNCTION CODE (f) DESIGNATOR

FORMAT III

35NVM060

Figure 8-5.—Illustrations of instruction word formats I, II, and III.

addressing designator **(i),** a 3-bit base designator or special selection code **(s),** and a 13-bit address displacement or operand designator **(y).** The **b** code (2^{19} through 2^{17}) is used to identify the index register (0-7) being used for indexing or operand address modification. The **i** code (2^{16}) is a ZERO when in direct addressing mode and a ONE when in indirect addressing mode. The **s** and **y** codes (2^{15} through 2^{0}) are combined to define one of the following: a 16-bit operand, a constant that can be modified by an index, a jump address, an indirect address, or a string of identifier bits.

- Formats IV-A and IV-B —The formats are for 16-bit or half-word instructions. These instructions reside in the upper or lower half-word of a memory location. They are normally stored two to a memory

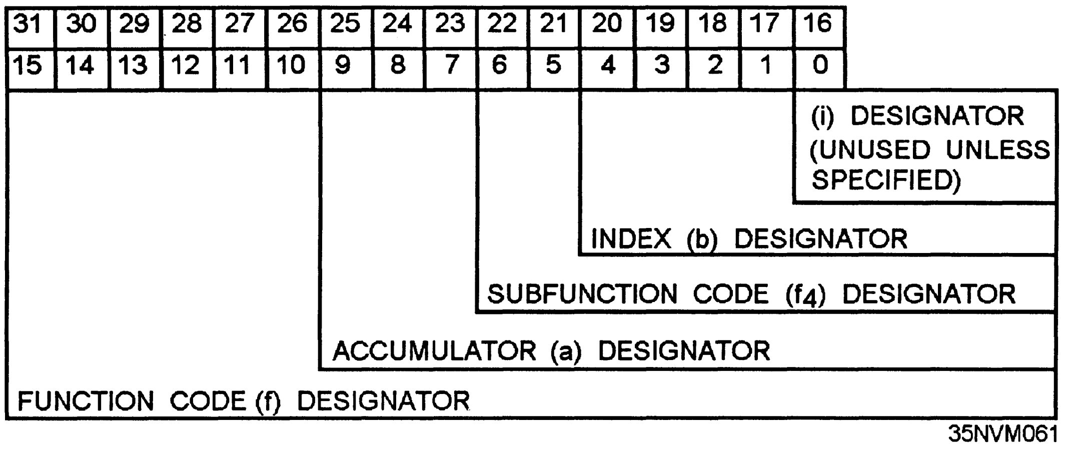

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
(i) DESIGNATOR (UNUSED UNLESS SPECIFIED)
INDEX (b) DESIGNATOR
SUBFUNCTION CODE (f4) DESIGNATOR
ACCUMULATOR (a) DESIGNATOR
FUNCTION CODE (f) DESIGNATOR
35NVM061

Figure 8-6.—Illustration of instruction word format IV-A.

word. First the computer executes the upper half-word instruction then the lower. If only one of these format instructions is to be stored in a memory word, then it is stored in the upper half-word location. An active status register (ASR) bit (2^{15}) is used to keep track of upper/lower half instruction execution.

Format IV-A instructions are used for a variety of computer operations that do not require an operand or operand address to be part of the instruction. These operations include but are not limited to mathematics and comparison operations, IOC commands, task and executive state operations, and real-time or monitor clock operations. The format IV-A instruction (fig. 8-6) is made up of an **f** field, **a** field, $\mathbf{f_4}$ field, an index designator **(b)** field, and **i** field, which is unused unless specified. The only field we have not covered is the $\mathbf{f_4}$ field, a 3-bit subfunction code. This field can be used to identify code memory registers (CMR) for CMR operations.

Format IV-B instructions are used to shift data stored in an accumulator. The accumulator designator specifies an accumulator in control memory. The shift count designator specifies a shift count or a source of a shift count. Instruction format IV-B (fig. 8-7) is made up of an **f** field, an **a** field, and a shift designator **(m)** field.

- Format IV-C —Format IV-C instructions (fig 8-8) are used for individual bit operations. These operations include setting, clearing, or testing an individual bit of a specified accumulator register. The 5-bit **n** field provides the bit position pointer to specify the register bit to be operated on.

- Format V —Format V instructions are full-word format instructions (fig. 8-9) used for single and double-precision floating-point math operations and other large magnitude number functions. In this format the **f,** $\mathbf{f_5}$**,** and $\mathbf{f_6}$ fields are used to define the specific operation to take place. The **a** and **b** fields are used for accumulator and index register definition. The **m** field provides decimal point positioning values for floating point operations.

INSTRUCTION OPERAND ADDRESSING

The types of operand addressing usually available are direct, extended, immediate, implicit, indexed, indirect, and relative.

Direct Operand Addressing

In direct operand addressing, the address of the operand's memory location is contained in the instruction. Figure 8-10 shows an example of direct addressing format.

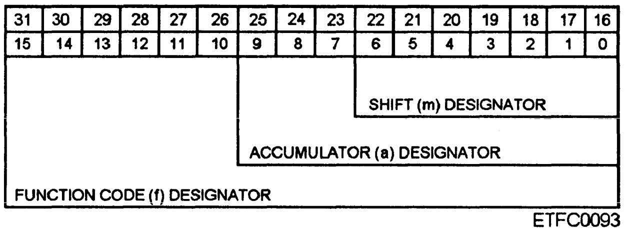


Figure 8-7.—Illustration of instruction word format IV-B.

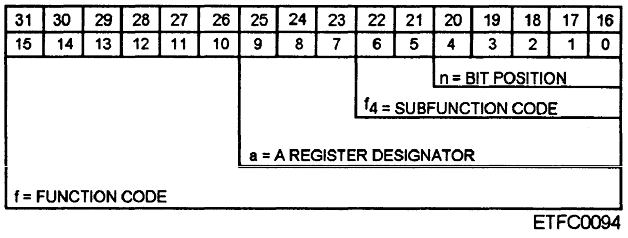


Figure 8-8.—Illustration of instruction word format IV-C.

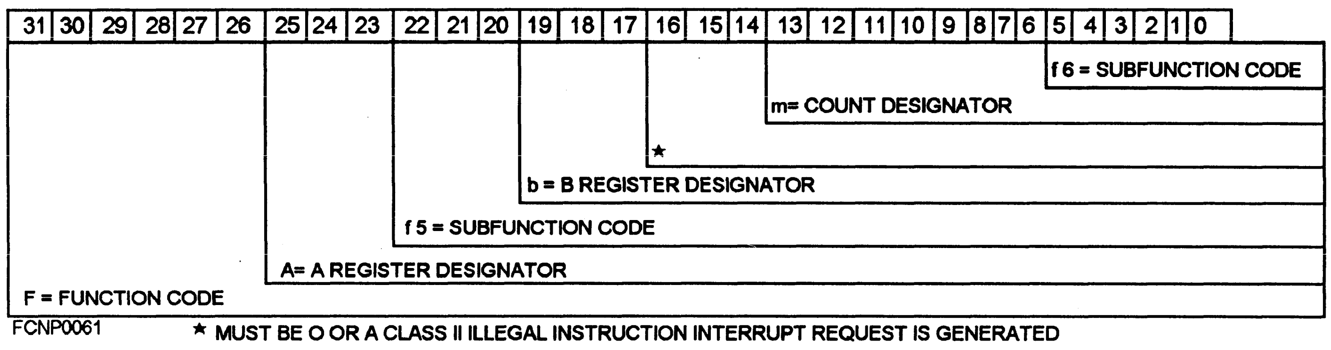


Figure 8-9.—Illustration of instruction word format V.

Extended Operand Addressing

Extended addressing. is used when an address of a memory location is too large to fit in one word. For example, on a computer with an 8-bit word (1 byte), only memory locations with addresses within the range of 0 through 255 can be addressed in 1 byte. To enable the computer to address memory locations with larger addresses, two bytes can be interpreted as one address. See figure 8-11.

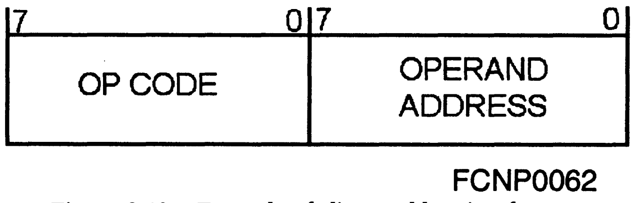


Figure 8-10.—Example of direct addressing format.

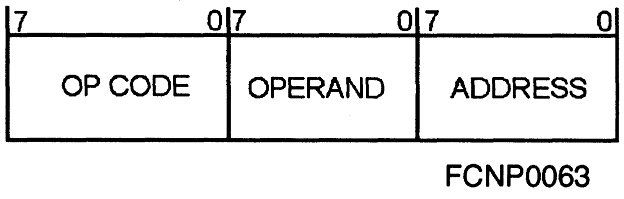


Figure 8-11.—Example of extended addressing format.

Immediate Operand Addressing

When the immediate format is used, the operand itself is contained in the instruction. In this instruction format, the destination is a general-purpose register defined by the destination register code (fields or designators) located in the instruction. Figure 8-12 is an example of immediate addressing.

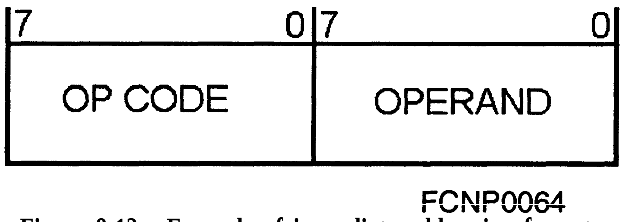


Figure 8-12.—Example of immediate addressing format.

Implicit (Implied) Operand Addressing

In implicit (implied) operand addressing, the operand location is implied by the op (function) code of the instruction (fig. 8-13). For example, the op code CLA could mean "clear the accumulator." No address needs to be specified because the op code contains all the information needed.

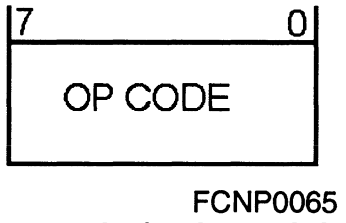


Figure 8-13.—Example of implicit (implied) addressing format.

Indexed Operand Addressing

In the indexed mode, the operand address must be generated when the instruction is being prepared for execution. This is done by adding the address given in the instruction to a value contained in a specified register. The register to be used is specified along with the operand address in the instruction. See figure 8-14. In this example, the parentheses are used to tell that the index mode is needed. The CPU will add the operand whose address is ADDR1 + the value in register 1, R_1,

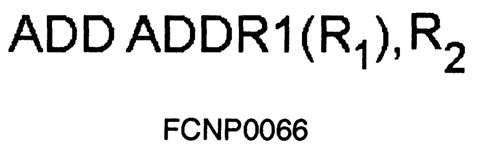


Figure 8-14.—Example of indexed addressing.

to the contents of register 2, R_2. By changing the value in R_1, different operands may be addressed. This is particularly useful for addressing memory locations in arrays. Indexing is a very useful troubleshooting tool. A short routine can be written to form a program loop to isolate on a specific malfunction.

On some computers, a CPU register is dedicated to this indexing function. In those cases, it is called an **index register** and is usually 3-bits or more depending on the computer type. Some computers permit a general-purpose register to be used as an index register.

Indirect Operand Addressing

Indirect addressing enables the operand address to vary during program execution by specifying a location in memory or a register in the instruction that tells where the address will be stored. See figure 8-15. In this example, the braces are used to tell that register 2 has been specified to hold the operand address. This means the contents of the main memory location whose address is contained in R_2 are added to the contents of R_1. Like the indexed mode of addressing, the indirect mode of addressing offers flexibility and is useful in addressing an array of data. Because the actual address pointing to an array can be stored separately from the program in memory, a large number of array pointers can be used.

Relative Operand Addressing

In many computers, particularly those with multiprogramming capabilities, a separate set of registers called base registers is used to define the start of particular blocks or segments of memory. Each block of memory could contain a separate application program. The contents of a base register is called the base address. Any execution of instructions or referencing of operands within the block of memory defined by the base requires that an offset or relative address be used. The offset is added to the base during instruction execution to allow reference of the correct instruction or operand address.

INSTRUCTION SIZE

Each address of memory (main or ROM) contains a fixed number of binary positions or bits. The number

$$\text{ADD}\ \{R_2\}, R_1$$

FCNP0067

Figure 8-15.—Example of indirect addressing.

of bits stored at a single address varies among types and generations of computers. For example, some store 8 bits (1 byte) at each location; others store 16, 32, or more bits at each location. The size of each memory location or **memory word** has a direct effect on the execution of machine instructions.

Basic instructions deal with **full word exchanges** as the register size is usually the same as the memory word size. In most computers, particularly those with large memory words, the capability exists to transfer less than a full memory word of information between memory and the applicable register. This allows memory words and registers to be further divided into economically sired bit groups for the most efficient use of memory for information storage and handling. For example, it is preferable to store two 8-bit characters in one 16-bit memory location than to waste an extra 16-bit location for the second character. Let's examine some of the various instruction sizes.

Full- or Single-Word Instructions

A full- or single-word instruction simply uses all the data contained in the instruction word to execute the instruction regardless of the size: 8-bit, 16-bit, and so on. Refer back to figures 8-3 and 8-5 for examples of full- or single-word instructions, 16-bit and 32-bit.

Half-Word Instructions

Half-word (upper or lower half) instructions consist of one-half of the normal instruction word size. The half-word instructions are executed by acquiring the complete normal instruction word, consisting of the half-word instruction to be executed and the next sequential instruction. After the first half-word instruction is executed, it is followed by the execution of the next sequential half-word instruction. If only one half-word instruction is used, it is usually located in the upper half of the instruction word with all zeros in the lower half of the instruction word. Refer back to figures 8-6 and 8-7 for examples of a half-word instruction.

Character-Addressable Instructions

In computers with word lengths greater than 8 bits, character-addressable instructions allow specified bit fields (called characters) of a word to be processed by the instruction. This is done in lieu of processing a whole-, half-, or quarter-word operand. Character addressing is permitted only when the instruction is executed in the indirect address mode. The particular operand bit field to be acquired is specified by the indirect word addressed by the instruction. In computers with an 8-bit word, no special instruction is needed because each character has its own address.

Double-Length Instructions

Double-length instructions consist of two adjacent words stored in memory.

Multiple-Word Instructions

Multiple-word instructions can be used to process two or more sequential words from memory. This concept is commonly used in microcomputers where the instruction word is 16 bits and the memory word size is 8 bits (a byte). In this case two or more sequential bytes from memory are transferred into two or more 16-bit registers for processing; or multiple word store instructions are used to process 16-bit registers into sequential bytes in memory (two bytes for each register). Refer back to figure 8-4 for an example of a multiple-word instruction format.

TOPIC 2—MAN/MACHINE INTERFACES

To use or maintain a computer, you must be able to control the computer's operation through some form of a **man/machine interface.** The man/machine interface is accomplished by the CPU and will vary with the type of computer. However, there are no major differences in the functions performed by the interfaces. You studied the controlling devices in chapter 3. The controlling devices allow you to interface with the computer. The methods are discussed in this topic.

The controlling devices used by operator and maintenance personnel vary with different types and generations of computers. In some cases the particular devices used are the same for both general system operation and the more specific maintenance functions. In many cases the man/machine interfaces have evolved from large panels containing many pushbutton/ indicators, and pushbutton/toggle switches, and switches (fig. 8-16) on a maintenance panel to more

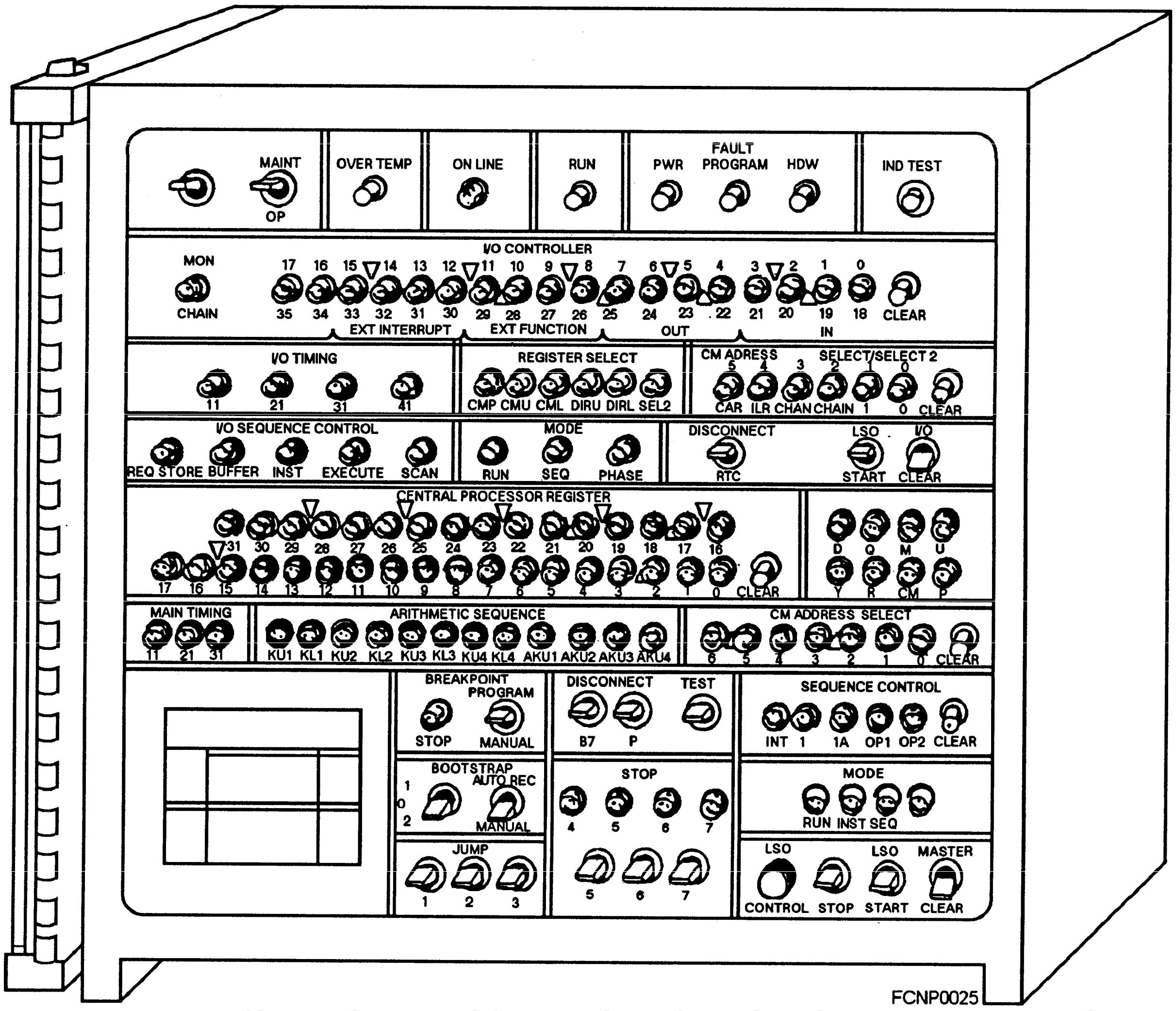


Figure 8-16.—Pushbutton/indicators, push button/toggle switches, and switches on a maintenance panel.

sophisticated microprocessor controlled assemblies containing display panels and data entry keyboards (fig. 8-17) on a display control unit (DCU).

In all cases, the man/machine interface provides you with some form of **data entry** and **data display** capability. The data entry function is used to enter commands or set parameters for computer operations, status, and test activities. The data entry can be made using the functions of the controlling devices. The data display capability is used to provide hardware status and other system description data to you. The data display capability can also allow you to react in some cases using menus to choose various operations. The man/machine interface is the primary path you use when requesting information on computer faults and for the computer to display the requested data.

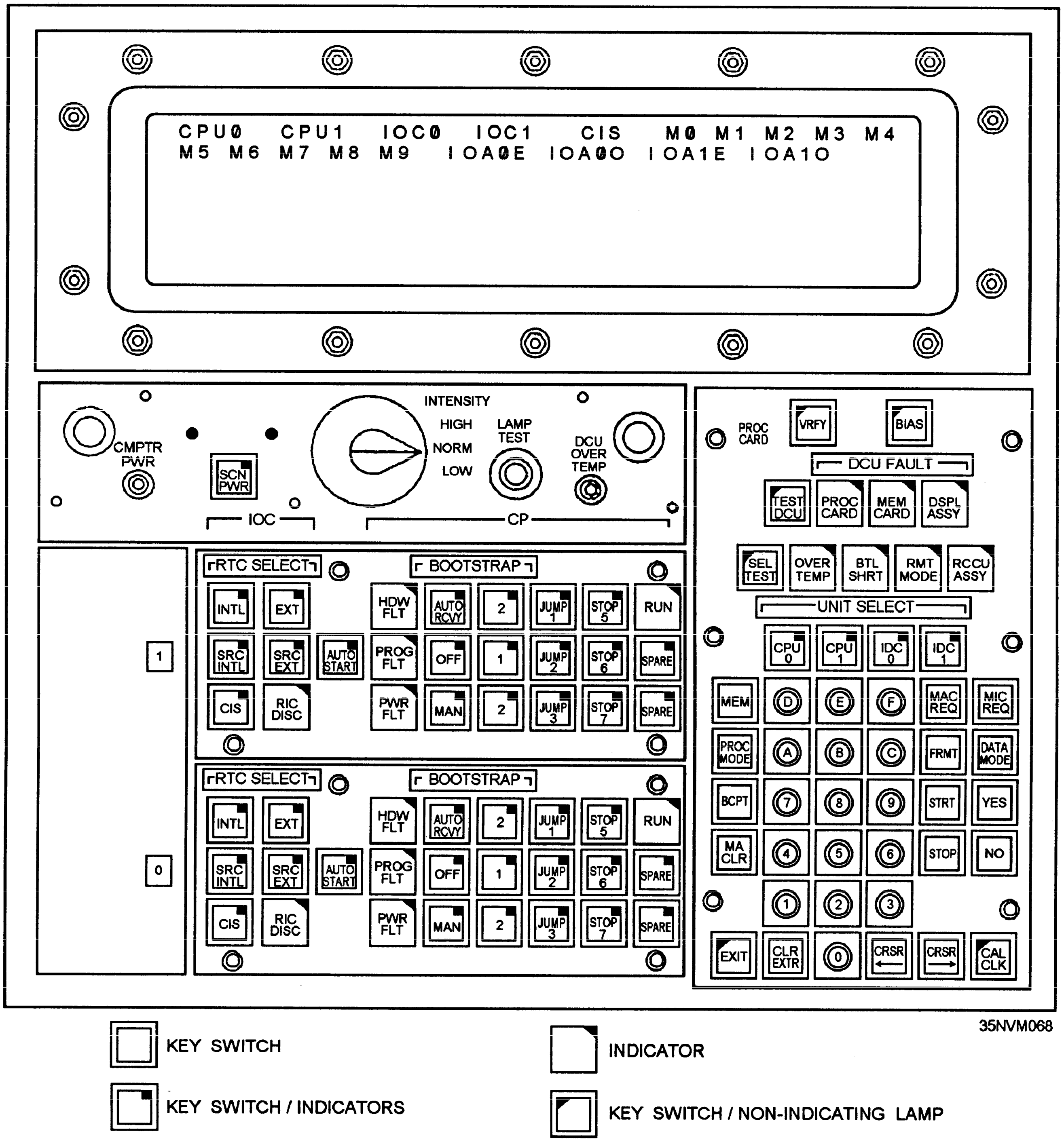


Figure 8-17.—Display panels and data entry keyboards on a display control unit (DCU).

MAN/MACHINE OPERATING MODES

Controlling the tempo of instructions through man/machine interfaces can be executed in several modes of operation. The two most commonly encountered operating modes are **run** and **stop.** Other modes are step, sequence, and phase.

Run Mode

When the computer is in run mode, it continually executes instructions one after another as directed by its logic circuits and software. The speed of execution is determined by the timing circuits or **clock** of the CPU.

Stop Mode

When the computer is stopped, it is not executing an instruction and will not execute an instruction until directed by an operator action **(START** or **RUN** pushbutton with the instruction address in the program counter). A running computer can be stopped by manual action (STOP pushbutton) or by execution of a STOP instruction under program control. Many microcomputers and embedded microprocessors do not have or do not use their STOP mode except from the device maintenance panel. During normal operation, they are designed to run continually from firmware programs once the equipment they are in is powered up. The only way to stop a microcomputer is to power it down.

Step Mode

Most computers or microprocessor controlled peripherals with maintenance panels offer the technician other modes of operations, specifically some form of **instruction step.** In the instruction step mode, individual instructions are executed one at a time as directed by the technician (pushbutton or toggle switch action) or in some machines at a slower than normal rate as determined by a manually adjustable low-speed oscillator. The contents of the computer registers and memory locations can be tested by the technician at the end of each instruction to verify proper operation or to aid in troubleshooting the computer. In newer computers, instruction step may be divided into two levels: **macro step** or **micro step.**

MACRO STEP.— A macro step allows the execution of a single macroinstruction. Those computers using macroinstructions composed of a series of micro instructions may give you the option to instruction step at either level, macro by macro or micro by micro within an individual microinstruction.

MICRO STEP.— A micro step allows the execution of a single microinstruction.

Sequence Mode

Sequence mode allows the execution of one sequence of an instruction at a time. Each operation of an instruction has an established set of sequences to complete the instruction. This enables you to execute one sequence of an instruction at a time. This is useful for detailed troubleshooting of an instruction.

Phase Mode

Phase mode allows the execution of one phase of an instruction at a time. If a computer has six main timing phases, you can execute one phase at a time. You can see what the instruction has accomplished at the end of each phase. This is also an aid for detailed troubleshooting.

MAN/MACHINE OPERATIONS

Interface capabilities available vary from computer to computer. Micros rely on keyboards and mouse devices to interface; consult your computer's manuals for detailed operations. Because more hardware is used on mini and mainframe computers, their interface capabilities provide a greater range to set parameters and control the operations of the computer more closely. This is particularly useful in the preventive and corrective maintenance aspects of your job. Without going into detail, the following functions are commonly available to the technician through the man/machine interface operating modes. Some are self-explanatory; we describe their basic operations.

- Master Clear —Clears all I/O and CPU registers and will stop the computer if it is in the run mode
- Start/Run —Starts the function determined by the operating mode(s)
- Stop (computer control) —Causes computer operations to stop
- Stop (program control) —Causes corresponding stops to occur under program control
- Jump —Causes corresponding jump to occur under program control
- Bootstrap —Addresses NDRO (ROM) depending on position of AUTO RECovery or MANUAL switches

• Real-time clock —Allows real-time clock to be updated internally or externally

Consult your technical manuals for exact operations used in the different computer operating modes.

MAN/MACHINE INTERFACE FUNCTIONS

The man/machine interface is used to perform a variety of general functions. These functions include, but are not limited, to the following:

- Configure the computer/processor system
- Apply power
- Enter data and display data
- Execute internal diagnostics
- Execute bootstrap
- Initiate operational programs
- Execute auto restart operations
- Execute diagnostics
- Patch or revise software

Not every man/machine interface function applies to every type of computer; therefore, we look at the three general types of computers (microcomputers, minicomputers, and mainframes) and give an overview of the man/machine interfaces used for each particular type as it applies to you. We do not address microprocessors as such. We consider them as replaceable or repairable components of the larger device. We also do not discuss peripheral devices used for system control and configuration operations. The following discussion covers only those man/machine interface devices considered as components or assemblies of the particular type of computer. With all types of computers, consult the appropriate documentation for your system to ensure proper operation. This last statement cannot be over emphasized.

Microcomputers

The man/machine interfaces used with the microcomputers you maintain will be system oriented. Let's take a look at the options available to you for microcomputers.

CONFIGURE THE PROCESSOR.— Microcomputer systems are designed to be flexible in their configuration. You can easily modify most desktop systems to incorporate additional disk units (hard or floppy), expanded memory, other components, as well as specific operator requirements. The ROM-based firmware that the system uses for booting the operating system as well as other system software must be configured for the current system interconnection scheme.

Three methods are commonly used to inform the processor of the system configuration. They are DIP switches, jumpers, and battery protected storage of configuration data.

DIP Switches.—Dual-inline package (DIP) switches are made to be instilled into integrated circuit sockets or board connections. Each switch in the package (fig. 8-18) normally indicates one of two conditions by its ON/OFF status. The board mounted DIP switches are designed so you can manually position them during component installation, removal, or initial system configuration to inform the processor of the availability of the particular components as well as the requirements of the system operators. They affect such operations as video display (color and resolution) and port(s) selections. Individual switches or combinations of two or three switches are used to specify a variety of configuration options.

Jumpers. —In some units, jumpers are used to make additional configuration changes. Jumpers (fig. 8-19) can be likened to dual-inline package (DIP)

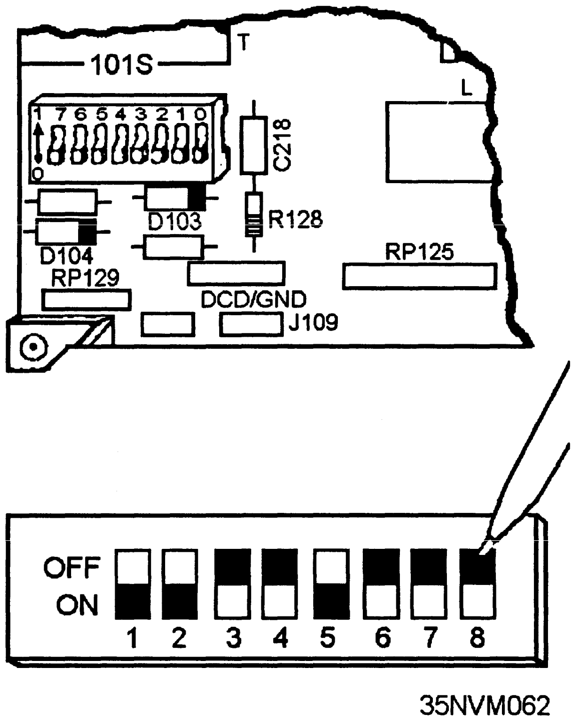


Figure 8-18.—DIP switches.

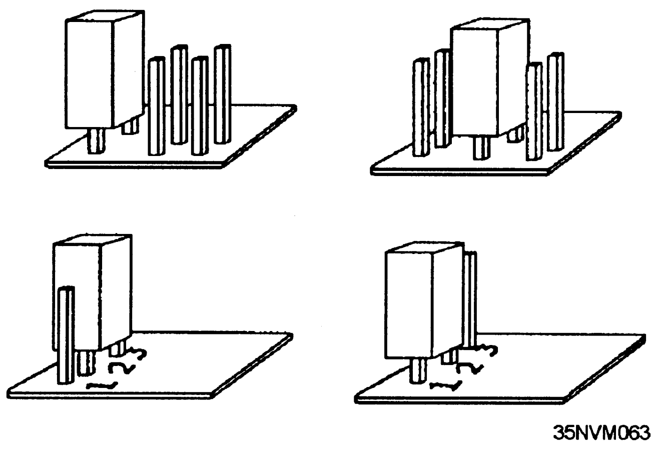


Figure 8-19.—Jumper connections.

switches except that you must physically remove and reinsert them. A jumper connector is designed for easy removal and reinsertion. They are permanent unless a configuration change is required. The jumper connector consists of a receptacle and plug arrangement. The receptacle is normally mounted permanently on the pcb's and/or backplane/ motherboard inside the micro's chassis. A plug (with or without a cable) makes the appropriate connection. It disables, enables, selects, and expands. Jumpers define the configuration of each pcb, which will eventually affect operations. Some of the functions affected include mode of operation (fast or normal), clock speed, wait states, and I/O connections. Like DIP switches, jumpers are designed so you can manually position them during component installation, removal, or initial system configuration to inform the processor of the availability of the particular components, as well as the requirements of the system operators. Individual jumpers or combinations of two or three jumpers are used to specify a variety of configuration options.

Battery Protected Storage.— Many newer microcomputers have a hardware setup/configuration program stored as firmware. It has the capability to display system configuration data on the display screen and to update system configuration data via the keyboard. The configuration data is stored in a random access memory (RAM) protected by a rechargeable battery so the data is retained for long time periods when the micro itself is powered down. The battery is located on the backplane/motherboard.

Configuration Options.— Both DIP switches and battery protected storage provide the same basic configuration data to the micro. System setup/ configuration options include the following:

- Date/time data (battery protected storage only)
- Base and expansion memory size
- Floppy disk drive identifiers (A, B, C or 0, 1, 2)
- Storage capabilities (number of Kbytes of storage per drive)
- Hard drive data
- Boot drive identifier
- Type of video display
- Video refresh time period

APPLY POWER.— Power is applied to the microcomputer with a simple ON/OFF switch usually mounted on the back of the desktop computer chassis (fig. 8-20). A separate monitor requires its own power switch. Portable micros usually have fixed time period rechargeable batteries (6, 8, or 12 hours) with a normal ac power option. Presence of system power is indicated by single indicator lamps on the front of the chassis and the monitor mounting. Sometimes in the same area as the ON/OFF switch, a selectable switch (fig. 8-20) called a **voltage or line select switch** allows the microcomputer to operate on voltages in the range of 100 to 130 volts or 200 to 230 volts.

USE CONTROLS, DATA ENTRY, AND DATA DISPLAY.— Micros, either portable or desktop

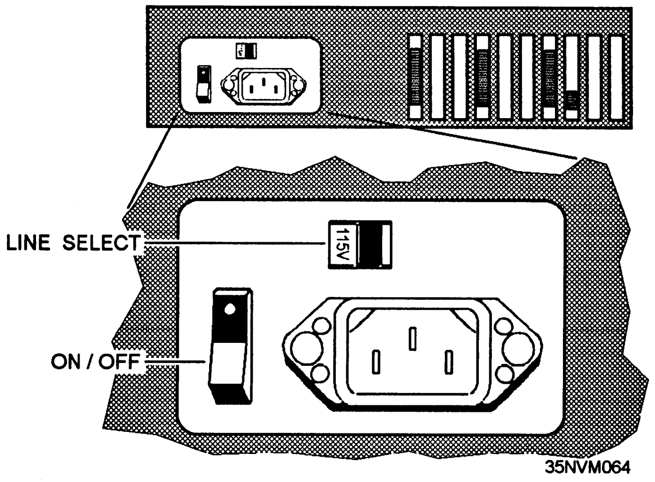


Figure 8-20.—Desktop computer back panel.

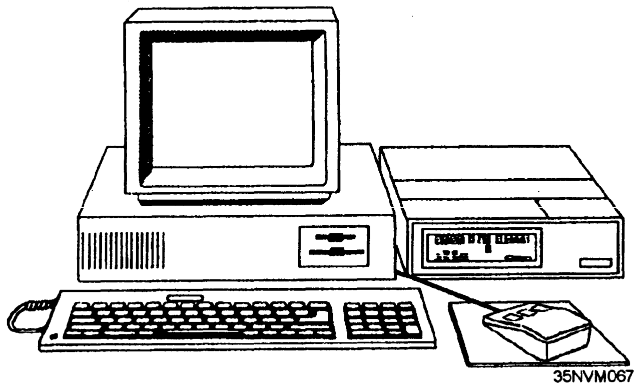


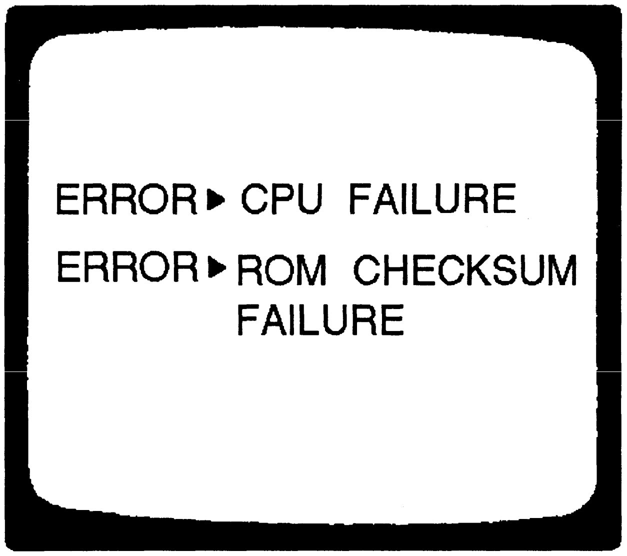
Figure 8-21.—A typical microcomputer with data entry and display devices.

(fig. 8-21), combine both maintenance and operational functions in the same data entry and display devices. Virtually all operator/technician commands are passed from the keyboard to the microprocessor. With the exception of a few simple indicator lamps, virtually all data is displayed on the monitor or display screen. Together the keyboard and monitor allow you to run software programs, perform tests, and view results. The keyboard and monitor on a microcomputer limit you to only data entry and display functions; there are no controls for power, cooling, or battle short conditions. With microcomputers, you can also use a mouse with the keyboard to interface with the computer.

EXECUTE INTERNAL DIAGNOSTICS.— As part of the power on sequence, microcomputers usually run a series of internal diagnostic programs. These are stored as firmware and take several seconds when the computer is turned ON. If everything is correct, the disk operating system (DOS) will load and the appropriate DOS displays will display. If there is a computer failure of any test, the computer ties to display an **error message** (fig. 8-22) on the display/monitor screen. Error messages identify the likely cause of the problem and possible solutions. Follow the recommended solutions closely and document the error message. If no error message is displayed or if the recommended solution does not fix the problem, more troubleshooting will be required. Most manuals will have a section that provides a detailed troubleshooting guide. The troubleshooting guide includes diagnostics that can be run from user selected tests available from the boot ROM program or disk based diagnostics.

Many micros are equipped with a more comprehensive set of internal diagnostics called **ROM-based diagnostics,** stored as firmware. These can be selected and executed using a special firmware controlled display. Some of these diagnostics are executed as part of the power on sequence, while others can only be executed from the display screen menu.

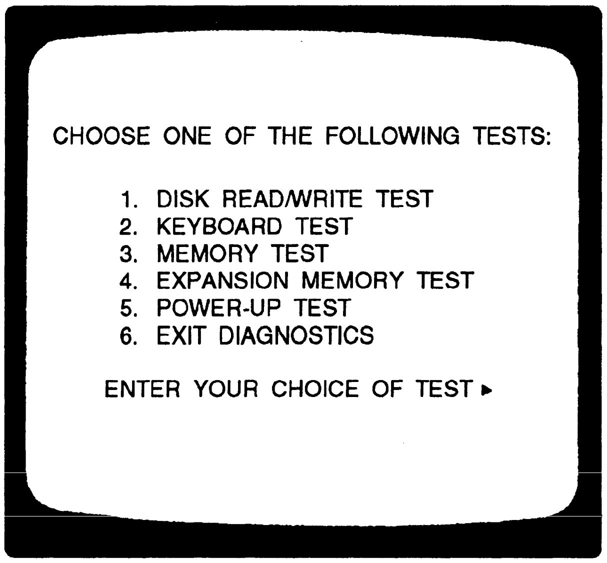
These diagnostics do not require any program loading. They are resident within the computer and accessible through a **menu driven display** (fig. 8-23). This enables you to select the desired diagnostic procedure and observe test status and error indications.



ERROR ▶ CPU FAILURE

ERROR ▶ ROM CHECKSUM FAILURE

Figure 8-22.—Example of an error message information.



CHOOSE ONE OF THE FOLLOWING TESTS:

1. DISK READ/WRITE TEST
2. KEYBOARD TEST
3. MEMORY TEST
4. EXPANSION MEMORY TEST
5. POWER-UP TEST
6. EXIT DIAGNOSTICS

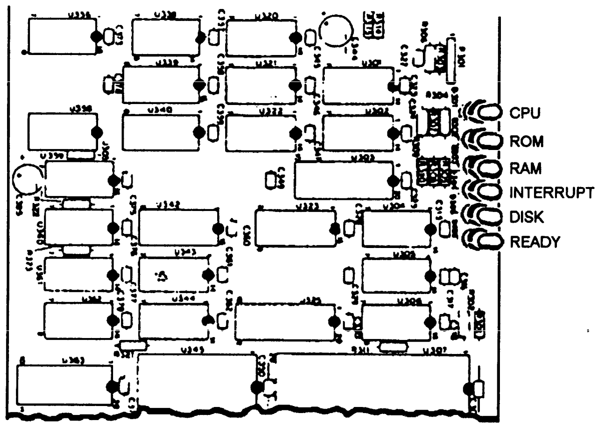
ENTER YOUR CHOICE OF TEST ▶

Figure 8-23.—Example of a menu driven display.

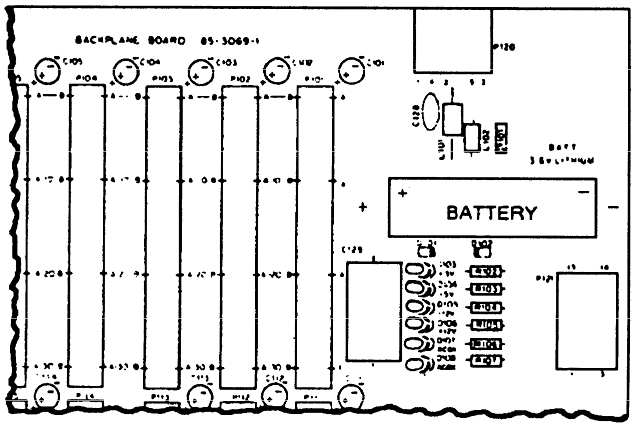
The ROM-based diagnostics menu provides you access to the following types of tests, again depending on the type of computer and the system configuration: disk read, keyboard, base memory, expansion memory, printer, and power on.

Additional board mounted diagnostic light-emitting diode (LED) indicators (fig. 8-24) are normally provided on the computer backplane and I/O logic circuit modules. This simplifies the diagnostic software and aids in fault isolation and identification. The LEDs on the backplane/motherboard for power remain on as long as the microcomputer is on. The LEDs on the I/O pcb extinguish as each test is successfully completed, except the READY LED. It will extinguish after an operating system is read from disk.

The features of ROM-based diagnostics of micros differ based on manufacturer and system configuration. They are normally designed to provide at least 90% resolution on detected faults to a single large scale integration (LSI) circuit or supporting integrated circuits. RAM and ROM errors are usually identifiable to the specific IC chip. The ROM-based diagnostics are designed to verify and fault isolate enough of the computer's logic to allow for loading and executing more comprehensive diagnostic programs stored on disk (floppy or hard disks).



A. I/O CARD LEDs (RED)



B. BACKPLANE LEDs (GREEN)

FCNP0072

Figure 8-24.—Examples of LED indicators.

EXECUTE BOOTSTRAP.— Micros are normally designed to **boot** or initially load the disk operating system (DOS) program from either the installed floppy or hard disk assemblies, based on the system configuration. The operating system program provides for operator control of the loading and executing of application programs used within the microcomputer system.

There are two ways to boot a micro. Firmware stored in PROM or ROM will automatically reference the configured disk for the operating system program as part of the power on sequence. Turn the micro ON and it automatically looks for the operating system program on the configured disk. If it finds it, the operating system automatically loads. If it does not find it, you will need to ensure the disks are setup correctly and depress a combination of keys to cause the system to boot.

INITIATE OPERATIONAL PROGRAMS.— For microcomputers, once the microcomputer has been booted, how the computer is configured will dictate how to initiate the operational program, the software, to be used.

EXECUTE AUTO RESTART OPERATIONS.— There is also a particular combination of keyboard keys (such as Ctrl, Alt, and Del) that will cause the operating system program to reboot and restart. This can be used in the event of a software failure. You can also reboot by turning the computer OFF and then ON.

EXECUTE DIAGNOSTICS.— You can load and execute **disk based diagnostics** using DOS command structures or a diagnostic monitor program. To execute these, you usually load the programs by a different power-up and boot sequence. The diagnostic monitor program displays a **test selection menu** similar to the internal diagnostic menu. Because these diagnostics are more comprehensive than the ROM-based diagnostics, you will be given more information on the menu than you are with the ROM driven display. The test selection menu provides for **diagnostic selection, test status,** and **error indications.** The selection, test status, and error indications are displayed on the microcomputer's monitor (fig. 8-25).

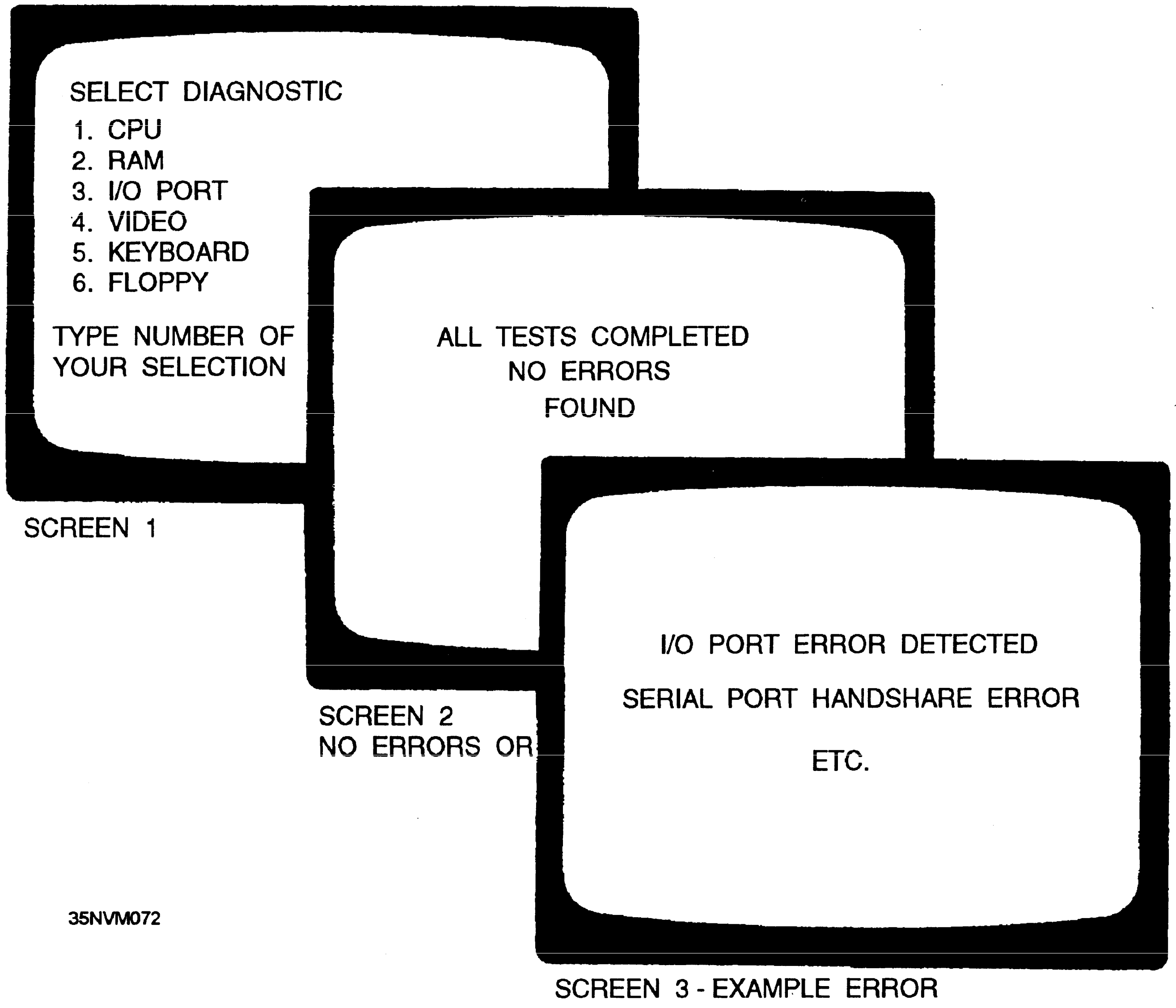
PATCH OR REVISE SOFTWARE.— While microcomputers have the same basic capabilities as larger computers, they are not designed to allow for the manual insertion and revision of machine code. At this time, revisions to operating system, application, or diagnostic software are provided by the system or software manufacturer or designer.

Minicomputers

The man/machine interfaces of the minicomputers you will maintain are more machine oriented and less system oriented.

CONFIGURE THE COMPUTER SYSTEM.— Minicomputers are primarily factory configured. There are a number of options you can incorporate by simply changing a module in the installed computer. As far as the computer itself, ensure that the controls and switches are set up properly for the intended operations. DIP switches and jumpers are also used in some minicomputers to meet the required interconnection scheme for the current system. In addition, make sure any peripherals or other equipments are configured correctly to ensure correct operation.

APPLY POWER.— Applying power to militarized minicomputers is somewhat more complicated than with commercially available micros. There can be a number of switches to power up the computer (fig. 8-26). Usually there is a remote panel that supplies power. Then at the unit itself there maybe a number of switches. Some use a circuit breaker that must be on before any of the other power switches will operate. Once the circuit breaker has been turned on,


SELECT DIAGNOSTIC
1. CPU
2. RAM
3. I/O PORT
4. VIDEO
5. KEYBOARD
6. FLOPPY

TYPE NUMBER OF
YOUR SELECTION

SCREEN 1

ALL TESTS COMPLETED
NO ERRORS
FOUND

SCREEN 2
NO ERRORS OR

I/O PORT ERROR DETECTED

SERIAL PORT HANDSHARE ERROR

ETC.

SCREEN 3 - EXAMPLE ERROR

Figure 8-25.—Examples of diagnostic selection, test status, and error indication displays.

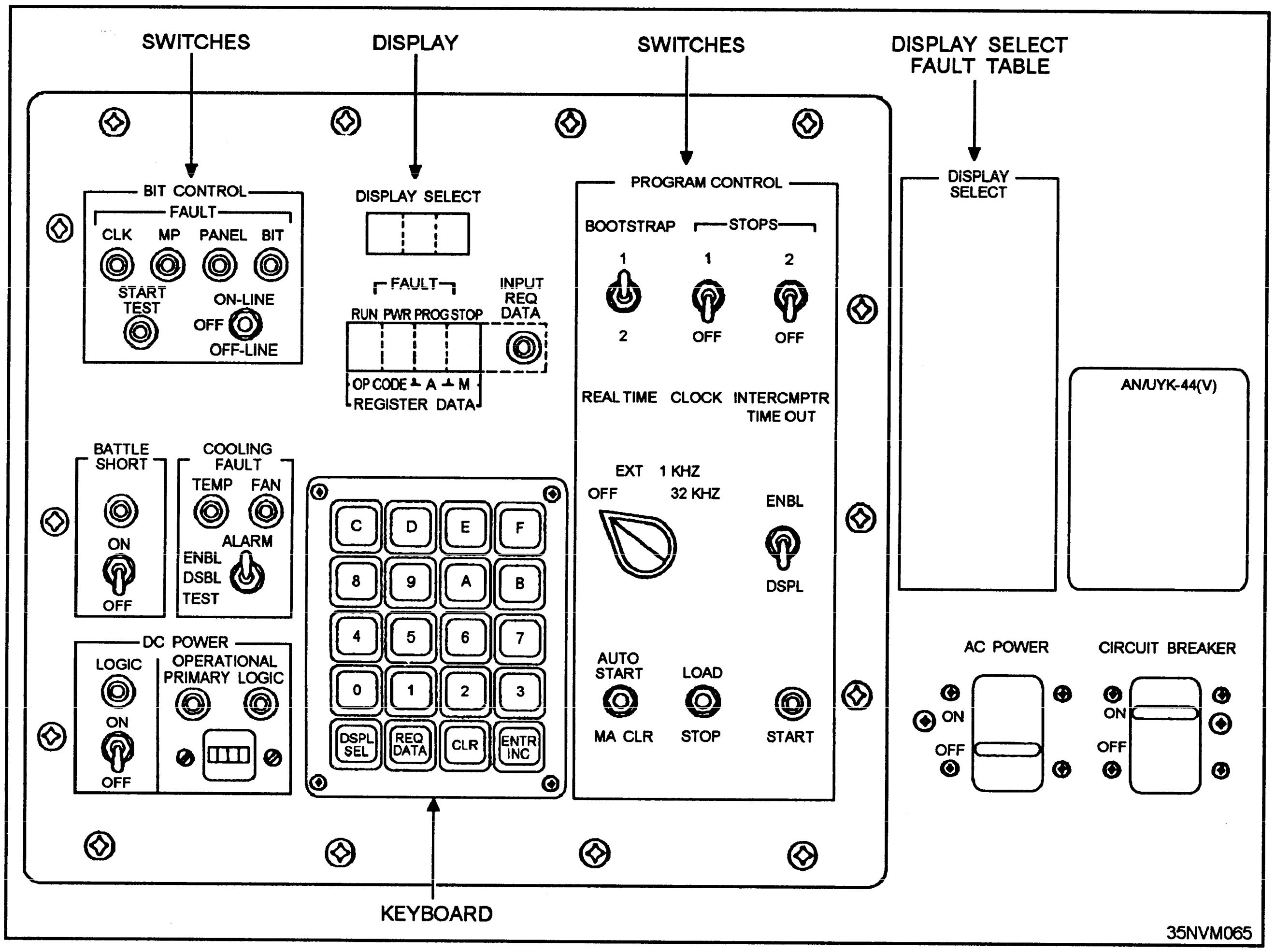


Figure 8-26.—Power up switches located on a maintenance control panel.

an ac power switch is activated to apply ac power to the computer. The circuit breaker will kick OFF in the event the computer power supplies draw excessive current.

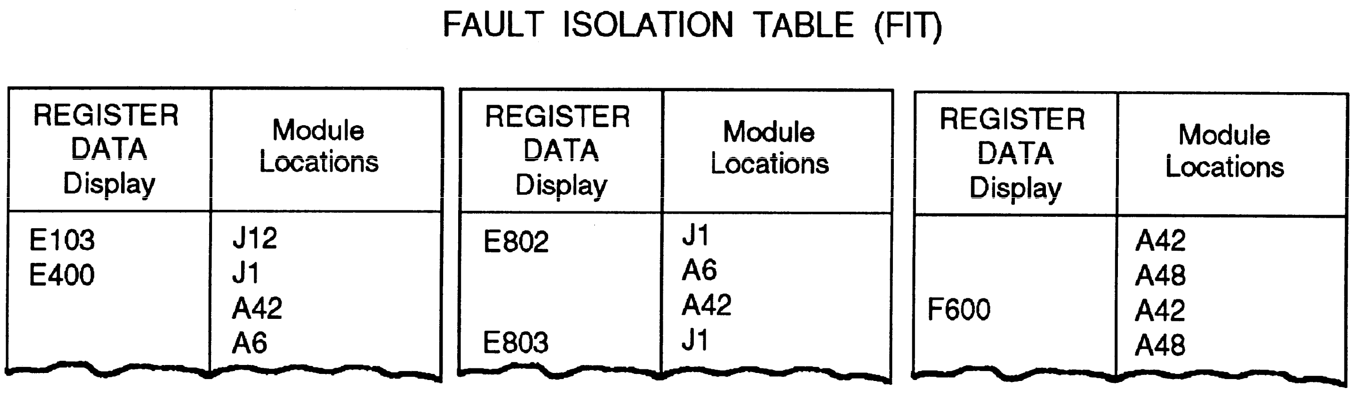
The ac switch allows ac voltage to be fed to the blower fans and dc power supply. Indicators, usually one for PRIMARY and one for LOGIC, show the presence of stable dc power when illuminated. Some minicomputers will have a 4-digit time meter to record the accumulated hours that logic power has been applied.

Some minicomputers are equipped with a battle short switch to allow the computer to run even when the temperature exceeds the normal allowable operating temperature established by the manufacturer. An audible alarm and/or indicators can also be used to indicate excessive temperature.

USE CONTROLS, DATA ENTRY, AND DATA DISPLAY.— The controls, data entry, and displays used on minicomputers vary. Some minicomputers have two panels, others one panel. When two are used, one panel is used for control and the other for maintenance. When one panel is used, the control and maintenance functions are located on the same panel. Refer again to figure 8-26. The panels on some minicomputers can be likened to the keyboard of a microcomputer; they deal primarily with the operating system and software programs. But with some minicomputers, you have more options. They include controls and indicators that deal with power and temperature. These two conditions were included in the apply power man/machine interface.

In addition to a number of control switches and indicator lamps, some minicomputers use a keyboard for data entry and numeric displays to show the contents of registers or display status. This is also illustrated in figure 8-26.

EXECUTE INTERNAL DIAGNOSTICS.— Internal diagnostics are built-in tests (BITs). Firmware and testing features are designed into the logic modules


FAULT ISOLATION TABLE (FIT)

REGISTER DATA Display	Module Locations	REGISTER DATA Display	Module Locations	REGISTER DATA Display	Module Locations
E103	J12	E802	J1		A42
E400	J1		A6		A48
	A42		A42	F600	A42
	A6	E803	J1		A48

35NVM070

Figure 8-27.—Example portion of a fault isolation table.

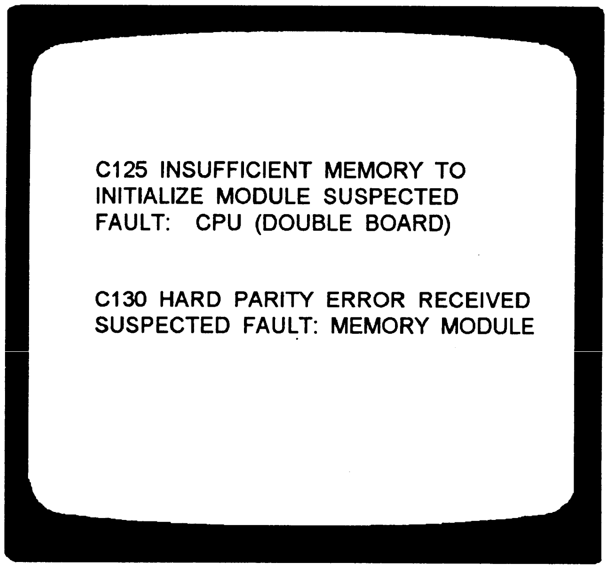
or an NDRO that can be executed at any time by the technician or operator. The BIT is designed to test the computer hardware (CPU, IOCs, and any optional circuits) and return pass/fail results to the operator/technician. Pass/fail results are displayed on the control, data entry, and data display man/machine interface. The BIT itself can consist of several levels of tests and subtests controlled from the computer's front panel. Some internal diagnostics are designed to test all or selected sections of the computer. Errors can be displayed on the front panel using the data display man/machine interface. The computer's technical manuals or a ready reference index located on the front panel will enable you to decipher the error code. A fault isolation table (FIT) lists the error code and the location of the recommended module(s) that will correct the problem. Figure 8-27 shows an example. On the pcb's in some minicomputers, LEDs are also used to aid in fault isolation and identification.

EXECUTE BOOTSTRAP.— Minicomputers are normally designed to boot or initially load the operating system program using a hardwired module (NDRO) located in the CPU. The NDRO is tailored at the factory and will select a particular peripheral device (disk, tape, and so forth) based on the position of the bootstrap switch located on the computer's controlling panel. Figure 8-26 shows a maintenance control panel with a bootstrap switch with two positions (1 or 2). The bootstrap program allows a more comprehensive program to be loaded from the selected peripheral into main memory and be executed. NDROs are also designed to perform a BIT, fault analysis program, or load a failure analysis program. To execute bootstrap, depress the run or load switch.

INITIATE OPERATIONAL PROGRAMS.— After the computer is booted, the operational program is loaded, initialized, and started. The operational program is tailored to meet the command's operational requirements or application.

EXECUTE AUTO RESTART OPERATIONS.— Auto restart operations are used when power is restored after a power loss.

EXECUTE DIAGNOSTICS.— Execution of external diagnostics can be loaded into the computer and controlled using an external control device. They can also be loaded into the computer from a peripheral (disk or magnetic tape unit) but initiated and controlled by the computer. These diagnostics are very thorough and also offer the option of testing all or specific sections of a computer. They are more comprehensive than the BITs. Figure 8-28 shows the test results of an external diagnostic test as they could be displayed on a controlling monitor.


```
C125 INSUFFICIENT MEMORY TO
INITIALIZE MODULE SUSPECTED
FAULT:  CPU (DOUBLE BOARD)

C130 HARD PARITY ERROR RECEIVED
SUSPECTED FAULT: MEMORY MODULE
```

ETFC0095

Figure 8-28.—Examples of test results from an external

PATCH OR REVISE SOFTWARE.— Minicomputers have the option to allow you to manually insert and make revisions to machine code or insert revisions using external peripheral devices. Patches or revisions to the software are written by authorized personnel only. The patches or revisions are entered using inspect and change routines or equivalents using the controls, data entry, and data display man/machine interface.

Mainframes

The mainframe computers used for tactical and tactical support data systems use a number of units and panels to control computer operations. Their controlling devices offer more options to perform the man/machine interface but their functions are the same.

CONFIGURE THE COMPUTER SYSTEM.— Mainframes are generally designed to work in large systems. In addition to a number of peripherals, they also work with major subsystems (display and communications). The software is designed to manage the computer and its resources based on the amount of hardware. Most large mainframe computer systems use two or more computers. This gives the system the capability to run in the event one of the computers goes down with hardware problems. Therefore, it is very important that you understand and know how to configure the system for full and reduced configurations. You accomplish this by knowing the capabilities and limitations of the software based on the quantity of hardware for your system and by ensuring all controls and switches on the computer(s), switchboard panels, and display and communication subsystems are correctly set.

APPLY POWER.— Applying power to mainframes also requires more than just turning on the ON/OFF switch. First, you must ensure there is power to the remote panel. Then at the unit itself, usually a circuit breaker must be applied, then blower and logic power. Indicators are usually provided for blower and logic to show there is stable power. Power to a mainframe is critical and you must ensure there is a stable power source. In addition to the circuit breaker protection, interrupts are generated if there are abnormal power fluctuations in which case the computer will shut itself down. Mainframes also use a 4-digit time meter to record the accumulated hours that logic power has been applied, except when there is a time meter for each module unit. Some mainframes have a separate power controlling device devoted entirely for power. It is usually on the front of the unit. Figure 8-29 is an example of a panel of such a device. It also monitors the temperature of the computer set.

Mainframes are also equipped with a **battle short switch** (also indicated on figure 8-29) and an audible alarm to allow the computer to run even when the temperature exceeds the normal allowable operating temperature established by the manufacturer and to indicate excessive temperature in the modules.

USE CONTROLS, DATA ENTRY, AND DATA DISPLAY.— Mainframes will use operator, maintenance panels, and/or display control consoles/units located near the unit. For our example,

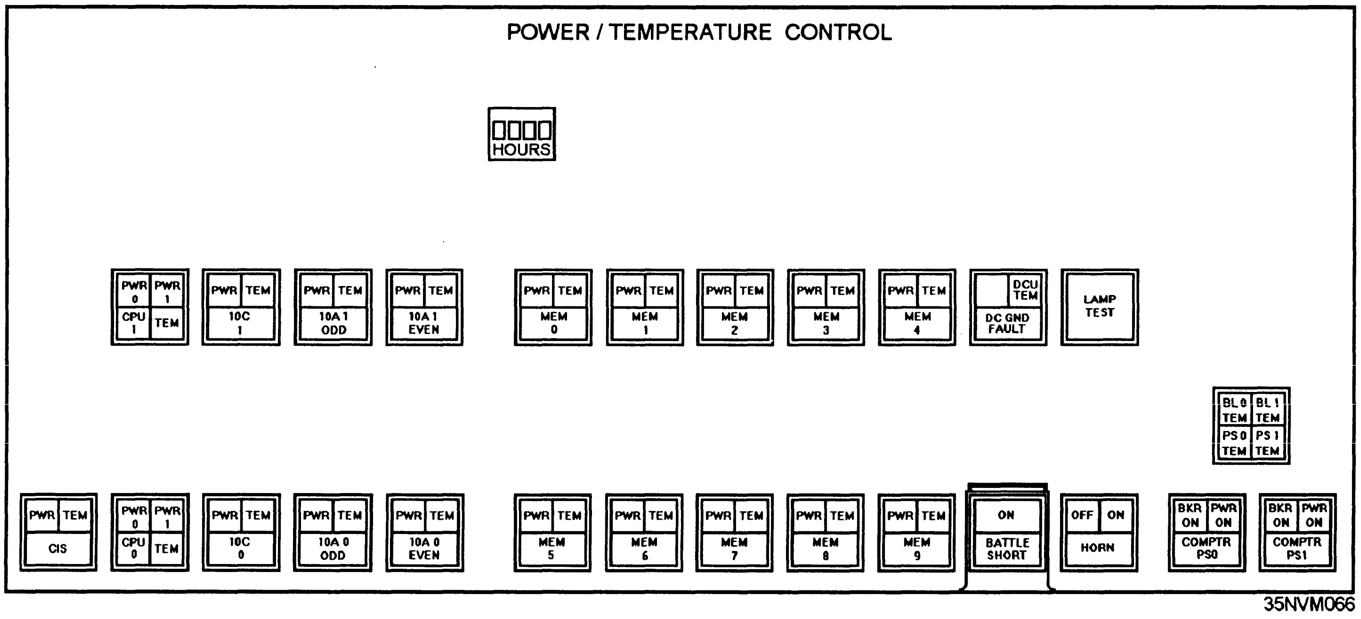
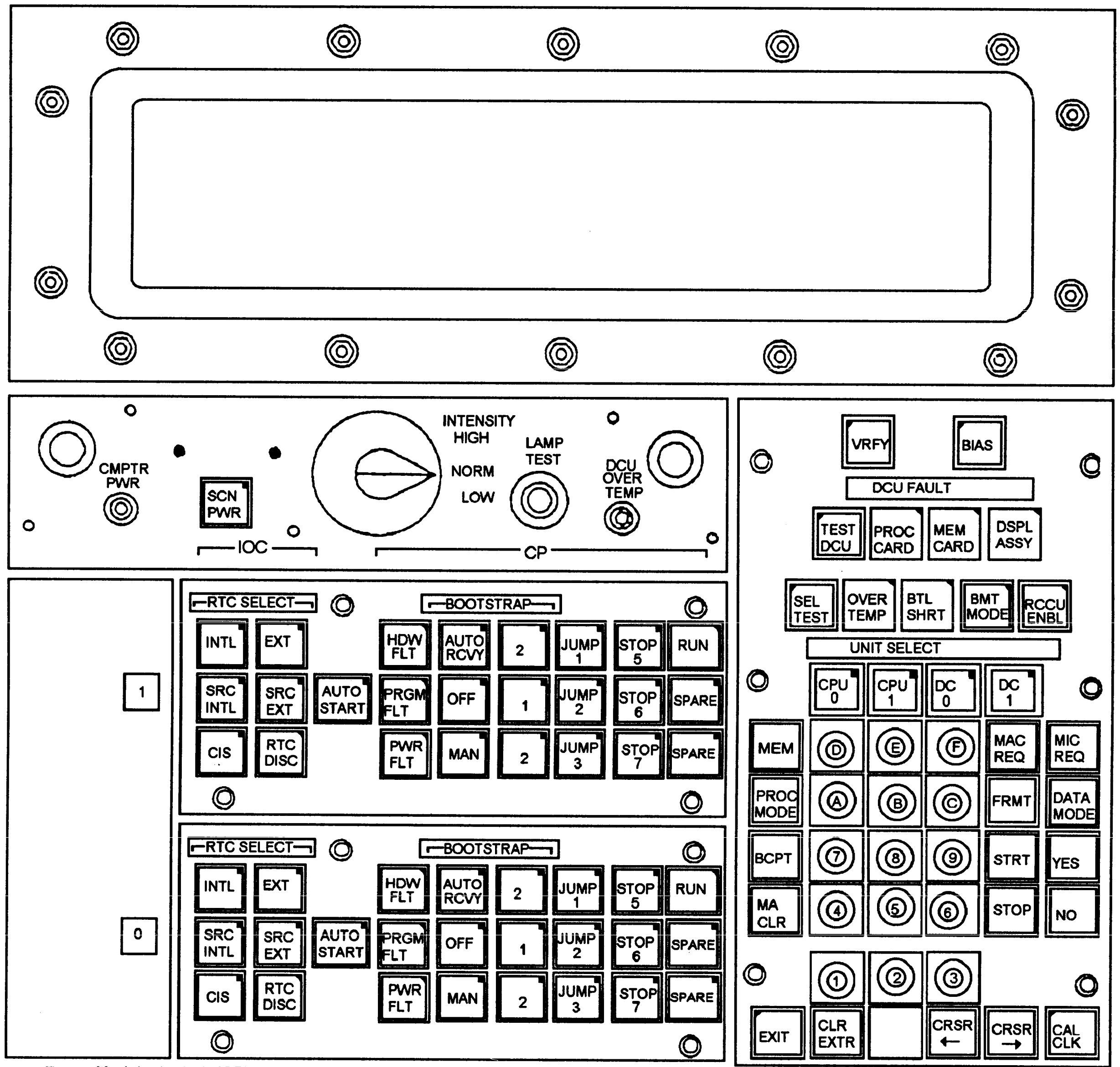


Figure 8-29.—Example of the panel of a power controlling device.

we show a display control unit (DCU) in figure 8-30. Remote units are also available to provide initial startup just like the operator and display control units. Control, data entry, and data display man/machine interfaces of mainframes are your primary means of operating and maintaining a mainframe computer. You can control all operations from this man/machine interface. Newer mainframes, in addition to controls, switches, and pushbutton indicators, use displays and keyboards to display status and to address the contents of registers.

EXECUTE INTERNAL DIAGNOSTICS.— On mainframes, internal diagnostics are also available using built-in tests (BITs) or tests available on an NDRO. They are designed to test the computer hardware (CPU, IOCs, and any optional circuits) and return pass/fail results to the operator. Pass/fail results are displayed on the control, data entry, and data display man/machine interface shown on figure 8-30. Similar to minicomputers, the BIT itself can consist of several levels of tests and subtests controlled from the computer's front panel. Some internal diagnostics are designed to test all or selected sections of the computer. Errors can be displayed on the front panel using the data display man/machine interface. The computer's technical manuals will enable you to decipher the error



T = Module tested (CPU#, IOC#, CIS - CPU#, MEM# - CPU#, 10A# E or O)
M = Module (CPU#, 10C#, CIS, MEM#, 10A# E or O)
L = LRU designation (A001-A014): 10A cards are designated as CHXX (octal)
A = Number of the LRU within the call out sequence
B = Total number of LRU'S in the call out sequence
CCC = Error code number

ETFC0096

Figure 8-30.—Example of a display control unit.

Table X-XX.—Diagnostic Error Codes		
OCTAL	HEXADECIMAL	TEST RESULT CODE (TEST ELEMENT LABEL)
052	2A	AM2910A Sequence Comparitor or Micro-I Register Parity Failed (ISEQUFD)-Suspect Cards MPC,CM
053	2B	ALU Failed Logical Functions (ILOGIC)-Suspect Cards CM,MPC
054	2C	ALU Failed Arithmetic Functions (IARITH)-Suspect Cards CM,MPC

Figure 8-31.—Example of diagnostic error codes.

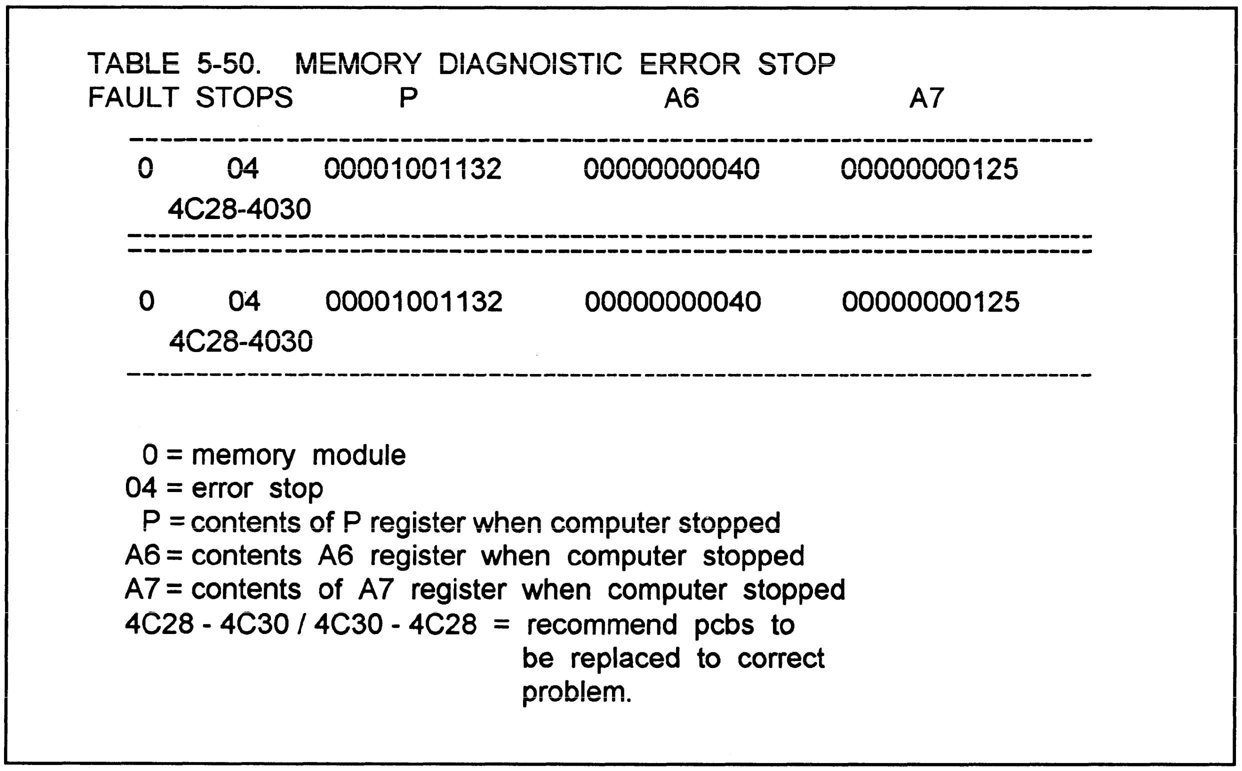
code. Figure 8-31 is an example. You can use this error code for fault analysis.

EXECUTE BOOTSTRAP.— Execute bootstrap works in a manner similar to the function on minicomputers. An NDRO is used to perform this function. The NDRO is tailored at the factory and will select a particular peripheral device (disk, tape, and so forth) based on the position of the bootstrap switches (0, 1, or 2) located on the computer's controlling panel (maintenance, control, display control, or remote unit). To execute bootstrap, select bootstrap switch 0, 1, or 2 and depress the start switch (fig. 8-30). NDROs on mainframes may also be designed to perform a variety of tests or other functions that may be selected by use of the DIP switches.

INITIATE OPERATIONAL PROGRAMS.— After the computer is booted, the operational program is loaded, initialized, and started. The operational program is tailored to meet the command's operational requirements or application. It is important that you know the software capabilities and limitations based on your hardware. Be sure your system is configured correctly.

EXECUTE AUTO RESTART OPERATIONS.— Auto restart operations are used when power is restored after a power loss.

EXECUTE DIAGNOSTICS.— External diagnostics can be loaded into the computer, executed, and controlled using an external control device. They can also be loaded into the computer from a Peripheral (disk or magnetic tape unit) but initiated and controlled by the computer. These diagnostics are very thorough. They offer the option of testing all or specific sections of a computer. They are more comprehensive than the BITs. Figure 8-32 shows an example of a defective card



```
TABLE 5-50.  MEMORY DIAGNOISTIC  ERROR  STOP
FAULT  STOPS          P                    A6                A7
    ----------------------------------------------------------------------
    0      04     00001001132      00000000040      00000000125
       4C28-4030
    ----------------------------------------------------------------------
    ----------------------------------------------------------------------
    0      04     00001001132      00000000040      00000000125
       4C28-4030
    ----------------------------------------------------------------------

     0 = memory  module
    04 = error  stop
     P = contents of P register when computer stopped
    A6 = contents  A6  register  when  computer  stopped
    A7 = contents  of  A7  register  when  computer  stopped
    4C28 - 4C30 / 4C30 - 4C28  =  recommend  pcbs  to
                                  be  replaced  to  correct
                                  problem.
```

ETFC0097

Figure 8-32.—Example of a defective card index (DCI).

index (DCI) with error stop and recommended corrective measures: Replace pcbs in locations 4C28-4C30.

PATCH OR REVISE SOFTWARE.— Mainframes also have the option to allow you to manually insert and make revisions of machine code or insert revisions using external peripheral devices. Patches or revisions to the software are written by authorized personnel only. The patches or revisions are entered using inspect and change routines or equivalents using the controls, data entry, and data display man/machine interface.

SUMMARY—COMPUTER INSTRUCTIONS AND MAN/MACHINE INTERFACES

In this chapter we introduced you to computer instructions and to ways you can interface with a computer. The following information summarizes important points you should have learned:

COMPUTER INSTRUCTIONS— Computer instructions are commands to the computer to tell the equipment to perform a designated operation. The instructions are processed by the central processing unit.

PROGRAMS.— Programs are sequences of instructions written for various purposes to solve problems or types of problems on a computer, to manage the computer's own resources and operations, and/or to maintain computers.

LEVELS OF INSTRUCTIONS.— Instructions may be either microinstruction or macroinstructions (a predetermined set of microinstruction).

INSTRUCTION TYPES.— Instructions may be classified by what they do, their operation. They may also be classified by their action on an operand-read, store, or replace.

INSTRUCTION SIZES.— Instruction sizes vary depending on the instruction and the computer.

INSTRUCTION FORMATS.— Every instruction has an operation (op) code to tell the computer what to do. It may also have an operand to give the address of the data to be operated on or to give other fields or designators.

INTERFACING WITH COMPUTERS.— The man/machine interfaces enable operators/technicians to control the computer's operation. These include control panels and operator panels/consoles.

MAN/MACHINE OPERATING MODES.— Computers can be operated in a variety of modes. This is very helpful when you are troubleshooting. Run mode continually executes instructions one after another. Stop mode causes the computer to stop; it will not restart until directed by some operator action. Step mode enables you to have the computer execute one instruction at a time so you can test the contents of computer registers and memory locations to verify correct operation or identify a problem.

MAN/MACHINE INTERFACE OPERATIONS.— Many operations can be accomplished by providing information to the computer through an interface.

MAN/MACHINE INTERFACE FUNCTIONS.— Many general functions can be performed through an interface.

It is up to you to learn all you can about how the computer systems you work with process instructions and what capabilities are available to you through man/machine interfaces. This will enable you to interpret computer instructions and interface with the computer to diagnose and isolate problems.